

AUGUST 1985

ÖS 55,– /sfr 6,50 Lit 5.500/hfl 8,–/dkr 30,– DM 6,50

Eine Markt & Technik Publikation

8|85 DAS MAGAZIN FÜR COMPUTER-FANS

Grafik

★ Die besten Programme
★ Marktübersichten
★ Grundlagen: Sprites und Hardcopies

Alles übers (Raub-) Kopieren

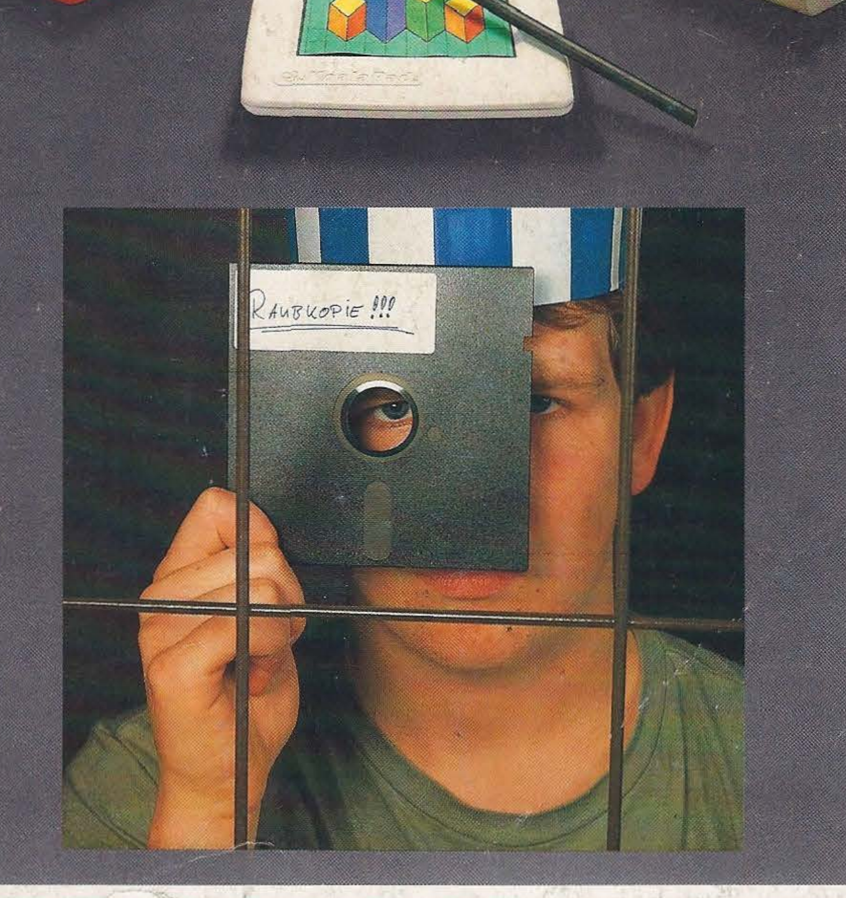
Der Sumpf II

Sprachen

★ Forth-Compiler zum Abtippen
★ Test: Profi-Pascal

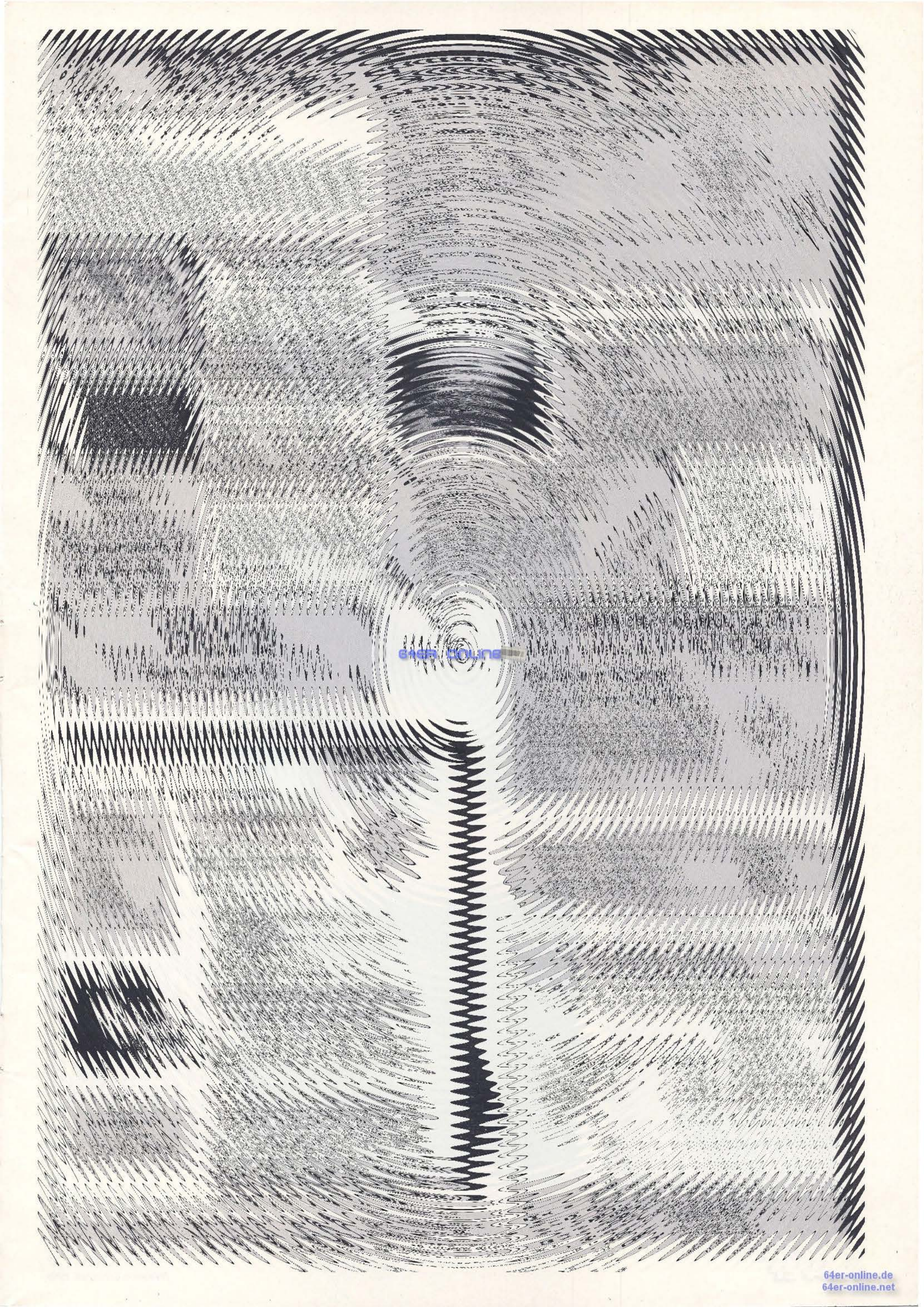
Kurs: C 64 extern

So beherrschen Sie den User-Port!



Tips und Tricks für C 64, VC 20 und C 16
Tolle Preise zu gewinnen ★ Das Neueste aus USA
★ Grundzüge der Schachprogrammierung
★ Schnelles Speichern mit Hypra-Save





Aktuell

Der Sumpf II

Grafik

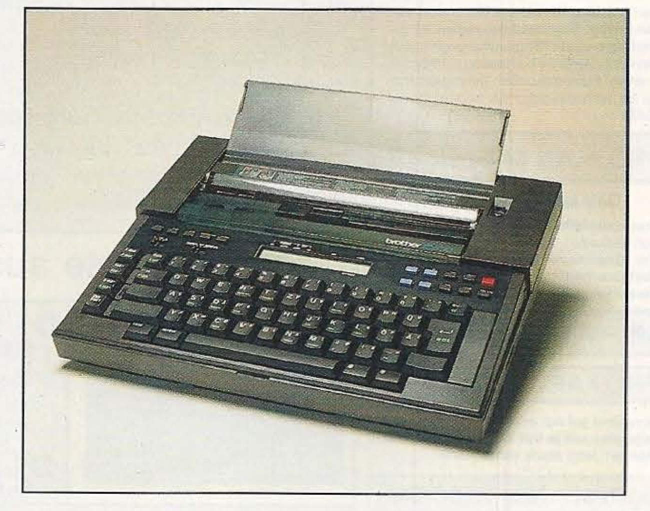
Hardware-Test

Hardware

Spiele-Test

Wettbewerbe

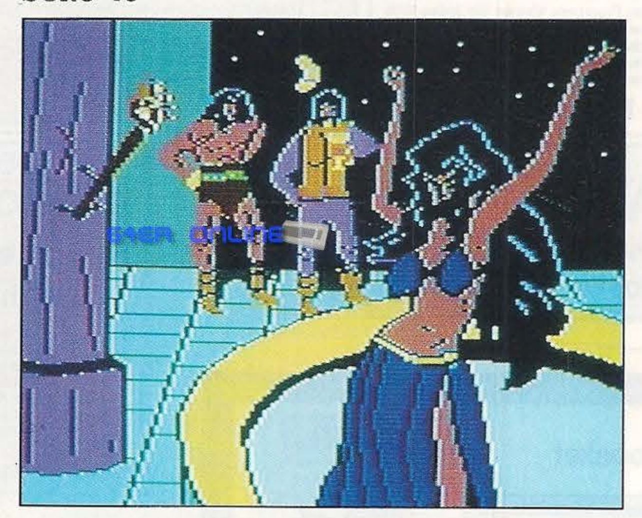
Seite 118



Der Handliche

Reisen Sie viel? Müssen Sie unterwegs Texte erfassen? Wenn Sie beide Fragen mit »ja« beantworten können, ist der Brother TC-600 für Sie interessant. Der TC-600 ist eine Schreibmaschine mit Textspeicher, ein Terminal zur DFÜ und ein Drucker für den Computer.
Seite 118

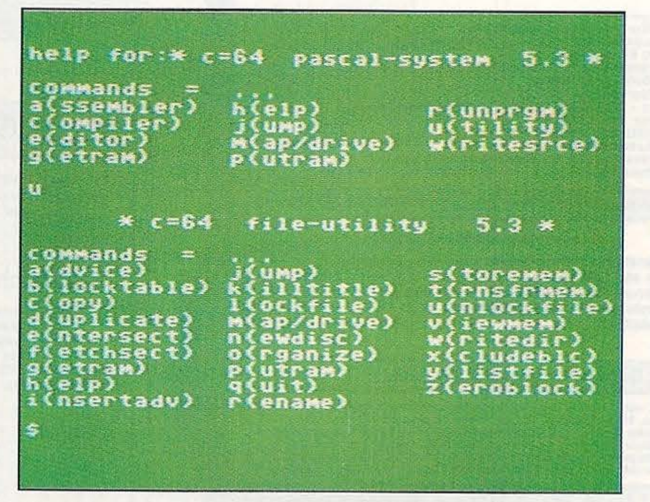
Seite 48



8 heiße Spiele im Test

Der Juni bereicherte den Spielemarkt um einige Neuerscheinungen, die sich durch ihre Qualität vom Durchschnitt deutlich abheben. Unter diesen Spielen sind Adventures mit sehr schöner Hires-Grafik und Geschicklichkeitsspiele mit viel Spielwitz.
Seite 48

Seite 122



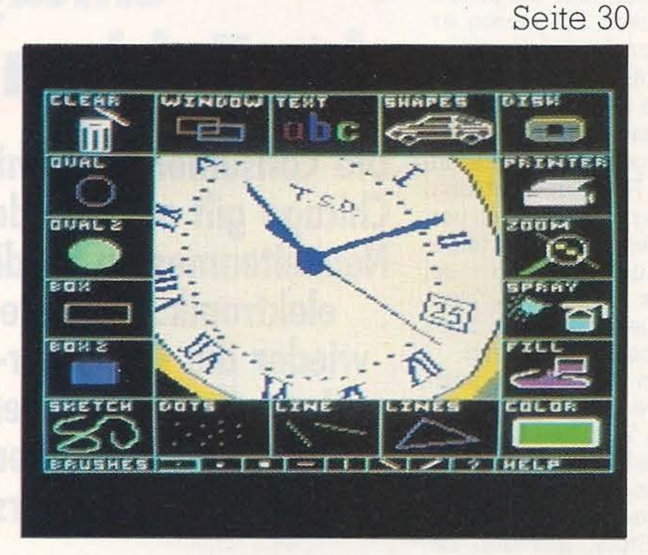
Test: Profi-Pascal

Das neue Profi-Pascal von Data Becker ist von uns genauer unter die Lupe genommen worden. Wir fanden ein Pascal, das den Wirth-Standard ohne Abstriche unterstützt und darüber hinaus noch zusätzliche Funktionen wie schneller Diskettenzugriff und relative Files zur Verfügung stellt.
Seite 122

Alles über Grafik

Wir geben Ihnen einen umfassenden Überblick der Grafik-Soft- und Hardware die es für den C 64 auf dem Markt zu kaufen gibt. Wir haben für Sie verschiedene Grafikprogramme getestet und sagen Ihnen, wo die Schwächen und Stärken liegen. In einem Grundsatzartikel erfahren Sie alles über die Spriteprogrammierung. **Seite 30**

Seite 30



Der Sumpf II

In dieser Serie sagen wir Ihnen, was beim Kopieren alles erlaubt ist und was nicht. Außerdem erfahren Sie einiges über den Wettlauf der Schützer gegen die »Knackies«. In einigen Hintergrund-Interviews nehmen bekannte und ehemalige Raubkopierer Stellung. **Seite 21**

Seite 21



Chicago im Zeichen der CES

Die »Consumer Electronics Show« stand diesesmal ganz unter dem Einfluß von Commodore. Der C 128 hat den Durchbruch in den Vereinigten Staaten bereits geschafft. Neue Software für dieses System ist angekündigt. Was es sonst noch gab, erfahren Sie auf Seite 8.

Seite 8



Listing zum Abtippen

Software-Test

Kurse

Rubriken



Sind Sie ein Räuber?

Wer ohne ausdrückliche Genehmigung des Autors auch nur für einen Bekannten ein Programm dupliziert, hat eine Raubkopie verfertigt — wenn das Programm dem Urheberrechtsschutz unterliegt. Das ist — so haben jetzt unsere obersten Richter entschieden — der Fall, wenn zur Erstellung der Software überdurchschnittliches Programmierkönnen erforderlich war. Ich halte diese BGH-Entscheidung inhaltlich für vernünftig: Es besteht kein Bedarf an besonderem Schutz für Software, die der Durchschnittsprogrammierer ohne besondere Schwierigkeiten schreiben kann — und für das, was im Niveau darunter liegt, erst recht nicht. Es besteht auch kein Schutzbedürfnis für Produkte, die nicht mehr dem Stand der Technik entsprechen.

Ausgesprochen praktisch ist die Entscheidung allerdings nicht: In Streitfällen müssen jetzt jedesmal Gutachter her, die beurteilen, was als Durchschnittskönnen anzusehen ist und ob das fragliche Programm davon genügend nach oben abweicht. Und da das »Durchschnittskönnen« keine gleichbleibende Größe ist, mit der man — wenn sie einmal festgelegt wurde — in Zukunft messen kann wie mit Meter oder Watt, dürften sich Streitfälle künftig länger hinziehen und zu höheren Gebührenrechnungen führen als bisher. Wo es um wichtige Entwicklungen geht, wird das kein Hindernis sein. In vielen Fällen von »Kleinkram« sollte das BGH-Urteil — und dieser Nebeneffekt ist nicht zu verachten — eine wünschenswerte Abschreckungswirkung haben: Mancher wird sich überlegen, ob und wie er bei einem xy-Produkt sein überdurchschnittliches Können nachweist.

Vereinfacht gesagt: Kopieren darf man, was sich eigentlich nicht zu kopieren lohnt. Die wirklich attraktiven Programme, hinter denen in der Regel ja auch überdurchschnittliches Programmierkönnen steckt, sind und bleiben geschützt.

Michael Pauly, Chefredakteur

Chicago im Zeichen der CES

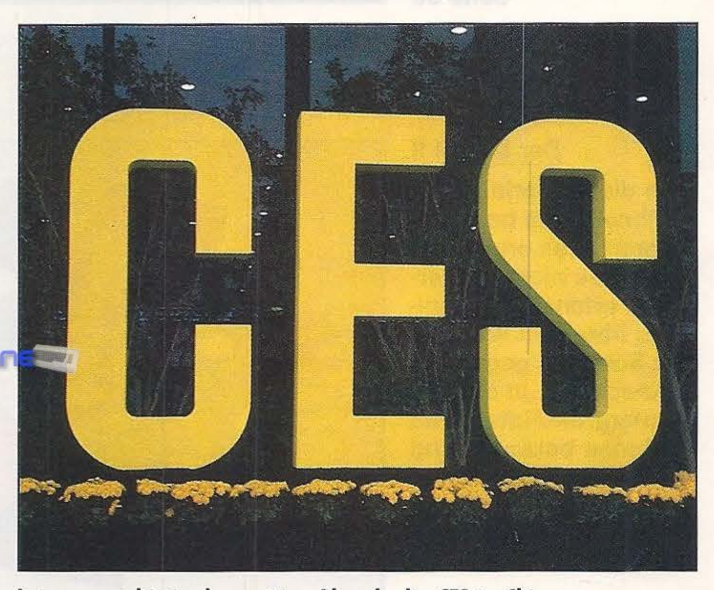
Die Consumer Electronics Show (CES) in Chicago gilt als eine der bedeutendsten Neuheitenmessen für die Unterhaltungselektronik. Vertreten waren auch wieder die Computer- und Software-Hersteller. Mit neuen, innovativen Ideen und Produkten will man die Gunst des Käufers gewinnen.

Selbst die größten Firmen mußten ihre noch vor einem halben Jahr aufgestellten positiven Prognosen erheblich nach unten revidieren. Erklärungen gab es soviele wie es wahrscheinlich auch Ursachen gibt. So wurde eine gewisse Marktsättigung bei Heim- und Personal Computern erkannt, ein vorsichtigeres und umsichtigeres Käuferverhalten sei ebenfalls zu beobachten, und schließlich sei da noch der ruinöse Wettbewerb.

Diesem Wettbewerb mußten einige Computerhersteller ihren Tribut zahlen. IBM stellte die Produktion des PC jr. ein und versucht durch Niedrigpreise die zirka 200000 Stück aus den Lagern zu bekommen. Apple wird drei seiner sechs Produktionsstätten in den USA und Übersee schließen sowie 1300 Mitarbeiter entlassen. Sinclair bekam nach den Schwierigkeiten, den QL im Markt zu plazieren, finanzielle Probleme und wurde schließlich von Maxwell aufgekauft. Atari war nur in einem kleinen Besprechungsraum auf der CES zu sehen. Der geplante Einführungstermin für den 520 ST im März konnte definitiv nicht eingehalten werden. Jack Tramiel will den 520 ST jetzt zuerst in Europa auf den Markt bringen. Es soll auch eine abgespeckte Version mit 256 KByte RAM geben. Die geplanten 5 Millionen Computer aus dem Hause Atari, wie Tramiel noch im Januar tönte, sind dadurch wohl zur Utopie geworden. Atari hat denn auch weniger Probleme mit der Technologie seiner neuen Computer als vielmehr mit der Finanzierung der Produktion.

Commodore auf dem Vormarsch

Einzig und allein Commodore scheint von den Strudeln noch relativ unberührt zu sein, obwohl auch sie mit sinkenden Verkaufszahlen zu kämpfen haben. Man ist sich aber ziemlich sicher, mit dem C 128 Personal Computer einen neuen »Millionseller« in der Produktlinie zu haben. 100000 Bestellungen seien schon eingegangen. Demzufolge konzentrieren sich auch nahezu alle Aktivitäten von Commodore auf dieses System. Die »Interimscomputer« C 116, C 16 und Plus/4 wurden vom Käufer nicht genügend akzeptiert (zu Recht, wie wir meinen).



Interessant bis in den späten Abend, die CES in Chicago

Für den C 128 soll es eine Reihe professioneller Peripherie geben. Neu vorgestellt wurde das Doppellaufwerk 1572 (Bild 1), das zwei horizontal eingebaute 5¼-Zoll-Laufwerke besitzt, zur 1571 und 1541 kompatibel ist, pro Laufwerk bis zu 410 KByte Speicherkapazität aufweist und ein schnelles Back-up erlaubt. Eingebaut sind des weiteren ein 6502 Mikroprozessor, 8 KByte RAM und 64 KByte ROM mit dem DOS. Die Übertragungsraten liegen bei 300 cps (characters per second) für den C 64 und jeweils 5200 cps (Burst rate) für den C 128- und CP/M-Modus. Das Laufwerk soll schreib-/lesekompatibel mit Kaypro, Osborne, IBM CP/M 86, Epson QX-10 und anderen Formaten sein. Der endgültige Preis wird erst bei der Markteinführung feststehen.

Zu sehen war auch bereits die 512-KByte-RAM-Erweiterung. Demonstriert wurden das »Nachladen« von Hires-Bildern: Eine ruckfrei, punktweise rotierende Erdkugel (pro Sekunde eine Umdrehung) war das Ergebnis. Anwendungen mit dieser Erweiterung lassen einiges erhoffen. Dieses Erweiterungsmodul kann von Basic 7.0 mit den Befehlen FETCH, STASH und SWAP wie ein Floppy-Laufwerk angesprochen werden. Die Übertragungsrate ist atemberaubend: 1 Million Byte pro Sekunde.

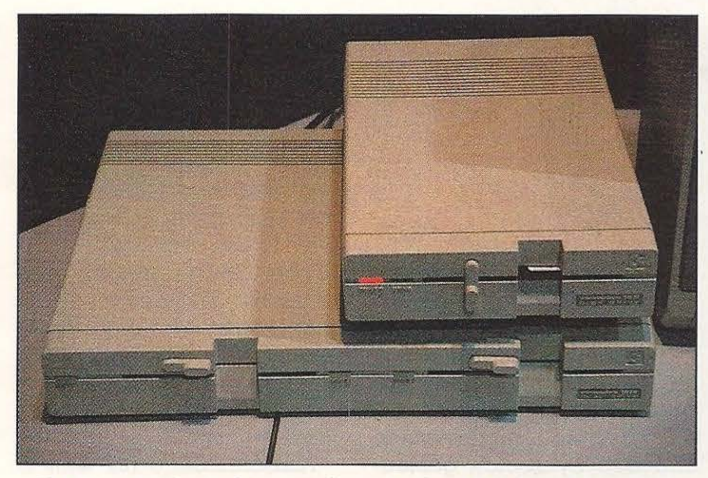


Bild 1. Die 1571 mit dem »großen« Bruder 1572 (unten)



Bild 2. Musik total für den Commodore 128

Ein neuer Matrixdrucker von Commodore konnte bewundert werden: der MPS 1000 mit aufgesetzter Traktorführung und NLQ-Schrift. Er hat allerdings eine verblüffende Ähnlichkeit mit dem GX-80 von Epson. Auch eine Maus wird es von Commodore für den C 128 geben. Extensiven Gebrauch von der Maus macht denn auch das kurz vor der Vollendung stehende Jane-Paket, bestehend aus janewrite, janecalc und janelist. Als nächstes integriertes Paket gibt es die unter CP/M lauffähige Perfect-Familie: Perfect Writer, Perfect Calc und Perfect Filer. Auf dem Commodore-Stand fanden sich auch einige Vertreter anderer Firmen ein, so etwa QRS mit einem Midi-Interface für 50 Dollar und über 200 Musikdisketten für den C 64 mit digitalisierten Stücken von Carmen bis Fly Robin Fly (Bild 2).

Da wir gerade bei der Software sind, soll gleich auf die neuen Programme der anderen Hersteller eingegangen werden. Auf dem Gebiet der Hardware hat sich diesmal bis auf wenige Ausnahmen die später noch beschrieben werden sollen, relativ wenig getan.

Neue Ideen – weniger Firmen

Eines hat sich ganz klar gezeigt: Die guten Ideen für neue Produkte sind den Software-Entwicklern nicht ausgegangen, nur deren Firmen haben Schwierigkeiten, die Programme an den Mann zu bringen. So war auch eine gewisse Konzentration bei einigen Herstellern zu erkennen. Besonders deutlich zeigte sich dies bei der Lernsoftware. Viele namhafte Firmen (Avant-Garde, CBS Software, DesignWare, EduWare, Electronics Arts, Grolier, Scholastic und andere) haben sich lediglich mit kleinen Teilständen auf einem Gemeinschaftsstand des größten US-Vertreibers Soft-Kat eingefunden. Synapse war auf dem Stand von Broderbund vertreten. Spinnaker dient als Überbegriff für Waltham Classics, Telarium, Better Working und Fisher Price. Doch lassen Sie uns zunächst die Neuigkeiten bei den einzelnen Software-Herstellern alphabetisch auflisten.

Access

Von Access gibt es jetzt den Nachfolger Beach Head II, der dort weitermachen soll, wo Beach Head aufgehört hat. Markerschütternde Schreie der tödlich getroffenen oder von Panzerketten zermalmten Soldaten wechseln sich mit Sätzen wie »I'm dead« oder »You can't hurt me« ab. Des weiteren wurde mit Mach 5 (35 Dollar) ein Modul für den Expansionsport vorgestellt, das die Ladegeschwindigkeit von der Diskette um den Faktor 5 erhöht und auch für den SX-64 geeignet ist.

Activision

Sechs Rohversionen von Programmen, die teilweise erst im Herbst verfügbar sein werden, waren bei Activision zu sehen. Besonders hervorzuheben ist die Idee, die hinter »Fast Tracks« steckt. Fast Tracks ist ein Autorennspiel mit eingebautem Generator. Soweit nichts Besonderes, davon gibt es bereits einige. Nur, neu gemachte Rennstrecken und der gesamte Spielablauf können auf einer extra Diskette abgespeichert werden und sollen an Freunde und Bekannte weitergegeben werden. Diese Verbreitung wird von Activision nicht untersagt, sondern sogar unterstützt. Eine Idee, die unserer Meinung nach Schule machen könnte.

Bei »Hacker« (Bild 3) taucht nur die Einschaltmeldung »Logon please...« auf, der Rest ist wie im richtigen Leben die Aufgabe des Anwenders. Ein Spiel voll im Zeichen der Zeit.

Weitere neue Titel von Activision sind GameMaker, Alter Ego, Computer Fireworks und There's someone living inside my Computer.

Batteries Included

Batteries Included arbeitet bereits an Versionen für den C 128 und Amiga. So wird es für den C 128 demnächst die Bestseller Pa-



Bild 3. »Hacker« von Activision. Betätigen Sie sich als »Informationssammler«.



Bild 4. »Goonies«, frei nach einem Film von Steven Spielberg

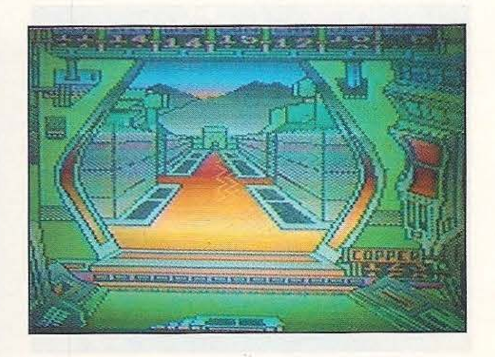


Bild 5. »Alternate Reality« von Datasoft. 3D-Grafik in Vollendung.



perClip, HomePak und The New Consultant geben. Dabei wird es nicht eine Diskette für den C 64 und eine für den C 128 geben, sondern beide Versionen sollen auf einer Diskette zum alten Preis angeboten werden. Für den Amiga will Batteries Included die »IS« Integrierte Softwareserie umschreiben.

Broderbund/Synapse

President Doug Carlston wunderte sich, wer die Mär vom toten Markt in die Welt gesetzt hätte. »Soweit wir betroffen sind, war der Markt noch nie lebendiger — für die richtigen Produkte«. Broderbund war bisher bekannt für hochwertige Arcade-Spiele, schaltete aber rechtzeitig um, und bietet nun Programme aus allen Gebieten an (The Print Shop, Bank Street Writer/Filer, Science Toolkit u.a.). Gerade diese haben zu einem guten Abschluß im ersten Quartal beigetragen. Für The Print Shop wurden drei neue Programme vorgestellt: The Print Shop Graphics Library Eins und Zwei sowie ein Companion. Aufsehen erregte auch Fantavision, ein Generator für »Special Effects«, vorerst nur auf dem Apple II. Synapse zeigte acht neue Programme, darunter Lode Runner's Rescue, Wizard of Wall Street, Brimstone, Essex und Mindwheel.

Cardco

S'more Basic ist der Name von dem in Deutschland entwickelten und in den USA durch Cardco vertriebenen Business Basic. 61183 Bytes free und über 60 neue Befehle sind die Merkmale von Business/S'more Basic.

Datasoft

The Goonies (Bild 4) ist eine Computeradaption des Films von Steven Spielberg. Die beiden Goonies-Kinder müssen dabei in jedem Bild zusammenarbeiten um vorwärtszukommen. Alternate Reality (Bild 5), ein Rollenspielabenteuer, überzeugt durch einen gelungenen 3D-Effekt und den unendlichen Tiefen der möglichen Kombinationen. Zorro, Elevator Action und Pole Position II sind weitere neue Produkte von Datasoft.

Epyx

Der Anziehungspunkt auf dem Stand von Epyx war Winter Games (Bild 6) mit sechs Wintersportarten wie Skispringen, Biathlon, Langlauf und Bobfahren. Von der Lucasfilm Games Division gibt es bei Epyx jetzt vier Titel: Ballblazer, Rescue of Fractalus sowie The Eidolon und Koronis Rift (Bild 7). Bei all diesen Spielen kann man sich die Gegend und die Schwierigkeitsgrade selbst auswählen.

Infocom

Diese, für ihre ausgefallenen Textabenteuer bekannte, Firma ließ sich wieder etwas Besonderes einfallen, um ihr neuestes Abenteuer »Wishbringer« einzuführen. Es wurde das ganze Naturkundemuseum von Chicago gemietet, inklusive Mumien und Personal. Es war wohl das erste Mal, daß man mit einem Sektglas in der Hand an Skeletten von Dinosauriern vorbeischlendern konnte. Wishbringer reiht sich denn auch nahtlos in die Reihe der Vorgänger ein. Die Faszination dieser Spiele ist an einem Beispiel besonders deutlich zu machen: Einmal auf den Geschmack gekommen, haben mehr als 50 Prozent vier, mehr als 18 Prozent acht Infocom-Adventures.

Mindscape

Einen ganz anderen Weg als Infocom beschreitet Mindscape bei seinen Abenteuerspielen. So benötigt »Deja Vu« (zunächst nur für den MacIntosh) keine einzige Eingabe über die Tastatur. Alles wird mit der Maus und den Symbolen gesteuert. »Racter« ist eine andere Art der Unterhaltungssoftware. Durch die komplette Sprachausgabe mit mehr als 2800 englischen Wörtern und der Beherrschung der englischen Grammatik ist Racter vorzüglich als Party-Entertainer geeignet. Versionen für den 128er und den Amiga sind angekündigt.

Eigenwillige Programme sind auch »The Luescher Profile«, ein Programm, mit dem Sie anhand von Farbkleksen Ihr Persönlichkeitsprofil erstellen können. »A View to a Kill« lehnt sich an die Handlung und den Namen des neuen James Bond-Filmes an. Ebenfalls neue Abenteuerspiele sind Forbidden Forest und Voodoo Island. Die Fantasie und das Verständnis für Delphine erfordert »The Dophin's Rune«, ein Spiel von einem kalifornischen Künstler entwickelt. The Halley Project ist ein Raumabenteuer in Realzeit.

Microprose

Microprose Software hat sich voll auf Simulationen spezialisiert. »Kennedy Approach« (Bild 8) versetzt Sie in die Lage eines Fluglotsen im Tower des JFK-Flughafens. Bis zu 20 ankommende und abfliegende Flugzeuge müssen dirigiert werden. Der Sprechfunk erfolgt dabei mit einer verständlichen Sprachausgabe. Mit »Silent Service« geht man zum ersten Mal in den Untergrund, genauer in ein U-Boot. Bei »F-15 Strike Eagle« dürfen wieder gegnerische Flugzeuge abgeschossen werden. Solo Flight und Hellcat Ace gibt es in verbesserten Versionen.

Sierra On-Line

Ausschließlich für den C 128 wird es von Sierra »Gato«, ein U-Boot-Simultionsprogramm geben. Damit will man nach Aussagen von Firmensprechern den Markt für diesen 128-KByte-Computer testen. Durch einen Vertrag mit Walt Disney Poductions gibt es zum ersten Mal seit sieben Jahren mit »The Black Cauldron« wieder ein Spiel, das auf einem Zeichentrickfilm von Walt Disney beruht.

Softsync

Speziell für den Kleinbetrieb bietet Softsync den »Accountant, Inc.« an. Für den deutschen Markt ist dieses Produkt sicherlich uninteressant, wichtig ist lediglich die Preisgestaltung. Bei nahezu gleichen Leistungsmerkmalen kostet Accountant für den C 64 ganze 50 Dollar, für den Apple IIc 100 Dollar und für den IBM PCjr 150 Dollar. Das heißt, je teurer der Computer, desto teurer auch die dafür angebotene Software. Eine Überlegung, die man beim Kauf mit berücksichtigen sollte.

Spinnaker

»Lernsoftware« auf einem neuen Medium gibt es von

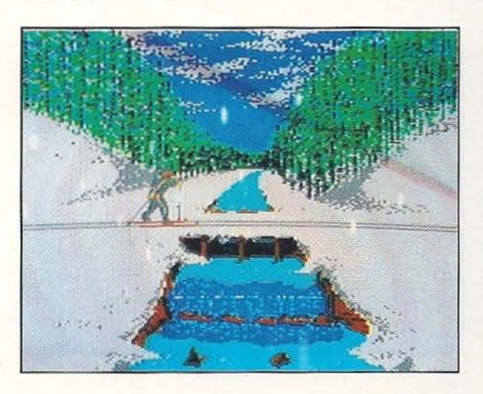


Bild 6. »Winter Games« von Epyx als ideale Ergänzung zu »Summer Games«



Bild 7. »Koronis Rift«, auf der Suche nach längst vergessenen Technologien

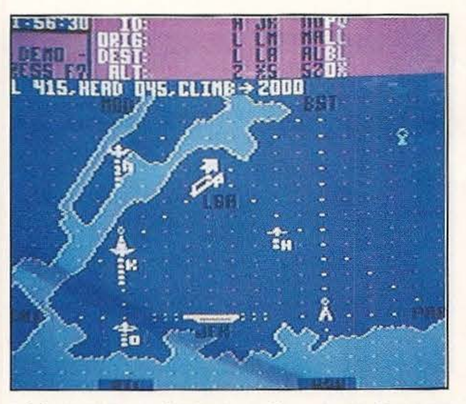


Bild 8. »Kennedy Approach«, gestreßter Lotse vom JFK in New York

Spinnaker. Videokassetten mit didaktisch aufgebautem Lernmaterial und Begleitbuch soll dem Lernenden den Zugang zur Rechtschreibung, Mathematik und dem Computer erleichtern. Windham Classics wartet mit den neuen Programmen »The Wizard of Oz« und »Below the Root« auf. Von Telarium gibt es die neuen Abenteuerspiele »The Case of the Mandarin Murder« (Bild 9) und »Nine Prices in Amber«. Better Working stellte die Heimproduktivitätsserie »Word Processor with Spellchecker« vor.

Springboard

Das überragende Produkt dieser Firma ist »Newsroom« (Bild 10), ein Programm, mit dem man sich seine eigene Zeitschrift erstellen kann. Newsroom gibt es momentan nur auf IBM und Apple, soll aber demnächst auf den C 128 umgeschrieben werden. Mit Newsroom können Sie den gesamten Ablauf einer Zeitung mitbestimmen und kreieren. Den Artikel und das Bildmaterial müssen Sie selber erstellen, Seitenumbruch und Layout werden vom Programm unterstützt. Datenfernübertragung von ganzen Seiten ist im Programm implementiert. Mit dem neuen »The Clip Art Collection« werden Ihnen bereits 600 Bilder zur Verfügung gestellt. Wurden neue Produkte genannt, die den Einsatz des Computers verändern werden, so waren Print Shop und Newsroom immer dabei.

Sublogic

Bekannt durch ihren Flight Simulator II, bietet Sublogic nun auch einen eher kriegerischen Flugsimulator an, »Jet« (Bild 11). Als Pilot einer F-16 oder F-18 können Sie mit modernster Bewaffnung ausgestattet, feindliche Jets abschießen. Für den Flight Simulator und Jet gibt es jetzt auch Landschaftsdisketten, die die verschiedensten Regionen der Vereinigten Staaten darstellen. Sublogic betonte, daß sie für die 68000-Computer ihre Programme umschreiben werden. Besonders freue man sich auf den Amiga von Commodore, der gerade für derartige Simulationen einen überragenden Geschwindigkeitsvorteil biete.

Timeworks

Auch Timeworks will für den C 128 Programme rausbringen, darunter »Word Writer 128 with Spell Checker«, »Data Manager 128«, und »Swift Calc 128 with Sideways«.

Schlußfolgerung

Es hat sich gezeigt, daß der C 128 auf dem amerikanischen Softwaremarkt bereits akzeptiert ist. Programme vom C 64 werden auf die neue 80-Zeichen-Darstellung umgeschrieben. Neue Produkte werden auf dieser Maschine angekündigt. Die Firmen gehen unterschiedliche Wege. Am sinnvollsten scheint dabei zu sein, beide Versionen, sowohl für den C 64 als auch für den C 128 auf einer Diskette unterzubringen. Der Amiga, obwohl von Commodore für diesen Computer eine noch nie dagewesene Abschottungspolitik betrieben wird, steht bereits bei vielen Software-Entwicklern. Die übereinstimmende Aussage von Leuten, die bereits mit dem Amiga gearbeitet haben sind: incredible, fantastic, unbeliefable... Es sollen sogar zeichentrickfilmähnliche Effekte auf diesem Computer möglich sein. Die offizielle Vorstellung des Amiga findet am 23. Juli in New York statt. Erst dann können wir genaueres über den Computer sagen. Sicher ist auf jeden Fall, daß es sich um eine außergewöhnliche Maschine handeln muß, der der MacIntosh und der 520 ST nicht das Wasser reichen können. Was es an Hardware Neues gab erfahren Sie in der nächsten Ausgabe. (aa)

Die Knacker-ROMs kommen

Gleich mehrere Hersteller bieten sogenannte Knacker- und Kopier-ROMs an. Auch neue Kopiersoftware gibt es.

Das GCS Cracker-ROM

Wie der Name schon sagt, handelt es sich hier um eine Kopierschutz-Knack-Hard- und Software. Das GCS Cracker-ROM ist eine Platine, die in den Betriebssystemsockel des C 64 eingesteckt wird. Das ROM ist eine Betriebssystem-Erweiterung, deren Hauptzweck dazu da ist, Programme zu knacken: Mit Space/Reset kommt man aus jedem Programm heraus, es existieren Tastenbelegungen, mit deren Hilfe man beispielsweise den nächstbesten Monitor von Diskette lädt. Module sind mit Hilfe des an den Kassettenport gesteckten An/Aus-Schalters start- oder speicherbar. Außer den Knackhilfen sind Programmierhilfen wie OLD (mit SYS 0) und ein DOS (mit SYS"Floppybefehl) eingebaut. Und zu guter Letzt ist auch Thomas Tempelmanns Fastload-Routine (ungefragterweise) enthalten. Das GCS Cracker-ROM könnte übrigens mehr, wenn der Autor die Programmteile vernünftig implementiert hätte.

Die Hardware ist ziemlich zerbrechlich: Schon beim ersten vorsichtigen Einsteckversuch brachen die Beinchen ab.

Data-Copy-Plus

Data-Copy-Plus ist als Kopiergerät für Datasette-Besitzer gedacht. Das Gerät wird an das C 64-Netzteil und zwei Datasetten angeschlossen. Die Übertragung geschieht digital, das heißt die Kopien erhalten eine bessere Qualität, als mit einem Doppeltapedeck kopiert. Die ideale Methode, Kassettensicherheitsbackups zu machen.

Original Data-Back-up-Box

Hierbei handelt es sich nicht um Knackhardware, sondern um ein 8-KByte-Modul mit Kopiersoftware.

Die Software ist ein 40-Spur-Kopierprogramm, das Read Errors mitkopiert (3 Minuten), ein 2 Minuten-Fast-Backup für ungeschützte Software, ein Filecopy mit 5facher Ladegeschwindigkeit sowie DOS-Kommandos und Directory-Anzeige. Die Herstellerfirma bittet um schriftliche Nachricht, falls ein Programm nicht damit kopierbar ist.

FCopy III

FCopy III ist das derzeit schnellste und sicherste Kopierprogramm, das es gibt. Das Programm ist nur für SpeedDos-Benutzer gedacht. Die Fähigkeiten des Programms sind: Kopieren einer ganzen Diskette in 40 bis 55 Sekunden inklusive Formatieren und Verify, Disk-Scan (Ansehen einer Disk mit Blockbelegung und Lesefehlern) in etwa 10 Sekunden. Beim Kopieren einer fehlerhaften Diskette kann man die Kopie vom Programm »reparieren« lassen, also die Checksumme neu berechnen lassen. Außerdem läßt sich die Anzahl der Schreib- und Leseversuche einstellen. Es kann die ganze Disk oder auch nur die belegten Blocks kopiert werden. (M. Kohlen/aa)



Bild 9. Aus dem Leben gegriffen: »The Case of the Mandarin Murder« von Telarium



Bild 10. »Newsroom« vorerst nur für IBM und Apple. Machen Sie Ihre eigene Zeitschrift.



Bild 11. »Jet« von Sublogic ist der verbesserte Nachfolger von »Flight Simulator II«



»Raub-Talkshow«

Im Rahmen von Seminaren und Symposien anläßlich der »C '85« konnten bei einer Talkshow zum Thema Software-Klau Betroffene mit Rechtsexperten diskutieren.

Die Teilnehmer dieser Talkshow setzten sich aus dem Münchener Anwalt Freiherr von Grafenreuth, dem Kölner Staatsanwalt Wolf, dem Leiter der Rechtsabteilung bei Commodore, Kröger, dem stellvertretenden Abteilungsleiter beim Kölner Jugendamt Greese und dem Vertreter der Gegenseite Wernéry, vom Hamburger Chaos Computer Club (CCC), zusammen. Moderator war Richard Kerler, Redaktionsdirektor beim Vogel-Verlag.

Nach den Einleitungssätzen wurde die Frage gestellt, warum denn so viele jugendliche »Kopierer« als Verbrecher verfolgt würden.

von Grafenreuth: »Es ist sehr schwierig, aufgrund von bekannt gewordenen Adressen von vorneherein festzustellen, ob es sich bei dieser Person um einen Jugendlichen, einen erwachsenen Computerfreak oder gar um eine GmbH handelt. Es hat sich gezeigt, daß das Gros der Raubkopierer zwischen 16 und 26 Jahre alt ist. Der jüngste in einem mir bekannt gewordenen Fall ist erst elf Jahre alt gewesen. Diese Kinder können allerdings weder strafrechtlich noch zivilrechtlich verfolgt werden. Diese Fälle müßte man dann dem Jugendamt übergeben. Das Mindestalter, um strafrechtlich vorgehen zu können, liegt bei 14 Jahren. Übeltäter, die dieses Alter erreicht haben, erhalten in der Regel eine Ermahnung oder eine Unterlassungserklärung zugeschickt. In bestimmten Fällen wird eine Hausdurchsuchung durchgeführt, bei der man dann Disketten und Computer beschlagnahmt. Meistens werden die Verfahren gegen Jugendliche aber vom Staatsanwalt eingestellt. Nur selten kommt es zu einer Anklageerhebung.«

Die Kinder bis 14 Jahre landen also beim Jugendamt, dessen Vertreter dann mit den Eltern Gespräche führen würden.

Greese: »Raubkopieren ist keine Frage der Erziehung sondern der Versuchung.«

Den Einwand, Raubkopieren sei eine Folge der überteuerten Programme, beantwortete Herr Kröger.

Kröger: »Die Entwicklungskosten für gute Programme sind sehr hoch. Dazu kämen oft noch erhebliche Lizenzgebühren, die Honorare für die Autoren, die Gewinnspannen für Händler und Hersteller.«

Bei der Bitte um konkrete Aufschlüsselung der Kosten bei einem Programm, das den Käufer 100 Mark kostet, konnte Kröger nur pauschale Antworten geben.

Kröger: »Bei einem Verkaufspreis von 100 Mark entfallen in der Regel 20 bis 40% auf die Handelsspanne, 10% gehen für das unternehmerische Risiko drauf, die Autoren erhalten in etwa 5 bis 10 Prozent pro Stück, der Gewinn für den Hersteller beträgt etwa 10%, dazu kommen noch die Entwicklungs- und Herstellungskosten in Höhe von 30 bis 55%. Diese Aussage muß man aber verallgemeinern. So kann die Kalkulationsgrundlage einzelner Unternehmen verschieden sein, und auch die Autorenhonorare können Pauschalbeträge beinhalten.«

Auf die Frage, wann denn nun Kopieren erlaubt sei, gab Herr Wolf folgende, erstaunliche Antwort.

Wolf: »Das Kopieren von Programmen für den persönlichen Gebrauch ist erlaubt. Es ist dabei unerheblich, ob man sich eine oder zehn Sicherheitskopien seines Originalprogrammes zieht. Die Weiterverbreitung dieser Kopien ist allerdings nicht erlaubt. Entfernt man einen angebrachten Kopierschutz, so ist dies ebenfalls zulässig, sofern die Kopien für den rein persönlichen Gebrauch bestimmt sind.«

Herr Wernéry brachte das Patentrecht ins Spiel.

Wernéry: »Mir ist bekannt, daß man ein durch ein Patent geschütztes Verfahren für den eigenen, privaten nichtgewerblichen Bereich nachbauen und verwenden darf. Warum sollte das also nicht auch bei den Kopien von Software der Fall sein?«

Daraufhin brachte Herr Kröger folgenden Einwand.

Kröger: »Ich weiß, daß zur Zeit etwa 1300 Verfahren für widerrechtliches Kopieren und Verbreiten anhängig sind. Geht man davon aus, daß jeder dieser Raubkopierer durchschnittlich 150 Programme in Umlauf gebracht hat, bei einem mittleren Wert von etwa 100 Mark pro Programm, so kann man sich leicht ausrechnen, welcher Schaden dadurch entstanden ist. Es geht im wesentlichen darum, die Spreu vom Weizen zu trennen, also die gewerblichen Raubkopierer von den einfachen Kopierern zu trennen. So ist es beispielsweise auch im Urheberrechtsgesetz erlaubt, sich ein Buch von der Bibliothek auszuleihen und vollständig oder teilweise, jedoch nur für den rein privaten Gebrauch, zu kopieren.«

Wolf: »Dies ist auch die Praxis der Staatsanwaltschaft, zumindest in Köln. Verfahren gegen Jugendliche, die nur kopieren, um ihre Sammlung zu vervollständigen, werden in der Regel eingestellt. Es soll keine Kriminalisierung dieser Jugendlichen vorgenommen werden. Auf der anderen Seite sieht man nicht immer, wer dahintersteckt.«

Um derartige Unsicherheiten zu umgehen, wurde eine »Gema-Lösung« auch für den Software-Bereich vorgeschlagen.

Wolf: »Eine Gema-Lösung wäre meiner Ansicht nach ein sinnvoller Ansatz. Die Software-Häuser wollen sich aber nicht abspeisen lassen.«

Kröger: »Diese Lösung fände durchaus den Zuspruch der Industrie. Vorstellbar wäre eine Verwertungsgesellschaft, die eine Schutzgebühr auf Leerdisketten erhebt und eine Abgabe an die Software-Firmen leistet.«

von Grafenreuth: »Auch nach meiner Meinung blockt die Industrie nicht. Dennoch sehe ich in einer solchen Verwertungsgesellschaft nicht die ideale Lösung. Was geschieht mit den Firmen, die nicht im Bundesverband der Software-Hersteller integriert sind? Probleme in dieser Richtung sind also noch genug vorhanden.«

Eine Zwischenfrage aus dem Publikum beschäftigte sich mit dem Problem der qualitativ mangelhaften Produkte einer bekannten Düsseldorfer Firma.

von Grafenreuth: »Wenn zugesicherte Eigenschaften von Programmen nicht vorhanden sind, so ist das ein zivilrechtliches Problem. Man hat die Möglichkeit, dagegen vorzugehen. Dennoch hat eine Klage wenig Sinn, da der Nachweis geführt werden muß, daß bestimmte zugesicherte Eigenschaften nicht erfüllt werden. Das ist auf seiten des Klägers mit oft nicht unerheblichen Kosten verbunden.«

Die Diskussion wurde noch einmal auf den Kernpunkt zurückgeführt, nämlich der Problematik, wann Kopien erlaubt sind und wann nicht.

Wolf: »Das Kopieren von Programmen ist erlaubt. Auch die Weitergabe von Kopien an enge Freunde ist nicht verboten (das ist vom Urheberrechtsgesetz abgedeckt). Es gibt allerdings unterschiedliche Auffassungen zu diesem Thema. Das ist oft Ländersache. Nicht erlaubt ist via Anzeige in Computerzeitschriften Tauschpartner oder Käufer für bestimmte Programme zu suchen.«

von Grafenreuth: »Die Verbreitung von Kopien im engen persönlichen Kreis ist durch das Gesetz abgedeckt. Die Weitergabe an gute Freunde: ja. An Bekannte zum Beispiel in der Schulklasse: nein.«

Die Frage eines ungarischen Kollegen, ob ein endgültiger Kopierschutz in Sicht sei, ergab verschiedene Reaktionen.

Wernéry: »Mir ist bis jetzt noch kein Kopierschutz bekannt, der nicht geknackt worden wäre.«

Kröger: »Die momentanen Kopierschutzmethoden sind sicherlich noch im Anfangsstadium. Dennoch wird die Zukunft möglicherweise Schutzmethoden hervorbringen, die nicht so leicht knackbar sind.«

Ein Zuhörer aus dem Publikum brachte noch einmal die Diskriminierung eines »unschuldig Verfolgten« in der Gesellschaft zur Diskussion.

Greese: »Die Jugendlichen (Kinder) sind durch die überraschende Verfolgung am Boden zerstört. Die Eltern werden durch die Aktivitäten ihrer Kinder überrascht. In der Umgebung einer an sich intakten Familie taucht plötzlich der Faktor der Kriminalisierung auf. Freunde und Bekannte distanzieren sich. Die Jugendlichen erfahren eine völlige Desorientierung. Man sollte alles daran setzen, eine Desorientierung der Jugendlichen zu vermeiden. Eine Lösung erscheint daher wirklich die Verwertungsgesellschaft zu sein.«

Wolf: »Als Staatsanwalt hat man sämtliche Seiten eines Falles zu berücksichtigen, sowohl die Raubkopierer als auch die Software-Hersteller. Die Interessen beider Seiten müssen abgewogen werden. Eine Hausdurchsuchung ist sicherlich ein wesentlicher Einschnitt im Leben einer intakten Familie. Man wird durch dieses Faktum in der Umgebung abgestempelt. Sicherlich ist eine Hausdurchsuchung nicht immer notwendig, aber wie soll man sonst die schwarzen von den weißen Schafen trennen? Die Staatsanwaltschaft sucht nach neuen Möglichkeiten.«

Ein Vertreter von Microsoft (Wordstar) äußerte die Ansicht, daß die Medien die Raubkopierszene zu sehr aufbauschen würden. Auch mit der Weitergabe von Kopien an gute Freunde zeigte er sich nicht einverstanden. Das würde dann nach dem Ostfriesenprinzip funktionieren: Ich kenne da einen, der hat einen Bekannten, dessen Freund ...

Ein anderer Einwand befaßte sich mit der Problematik des Anteils von Fremdteilen in eigenen Programmen.

Freiherr von Grafenreuth verglich diesen Sachverhalt mit der Literatur.

von Grafenreuth: »Wenn Sie in einem bestimmten Werk von Hemmingway den Schwertfisch durch Walfisch ersetzen, können Sie sicherlich noch keine urheberrechtlichen Ansprüche auf das neue »Werk« erheben. Ebenso verhält es sich bei der kompletten Übernahme von ganzen Kapiteln (siehe Anmerkung der Redaktion)«.

Als nächstes wurden die hardwarezerstörenden Schutzmaßnahmen angesprochen.

Wolf: »Die rechtliche Seite dieser Vorgehensweise wird noch geprüft. Ich sehe allerdings keine Bedenken, warum das nicht erlaubt sein sollte.«

Kröger: »Ich möchte hier noch einmal betonen, daß derartige Diskussionen in der Öffentlichkeit und in den Medien hoffentlich dazu beitragen, ein bestimmtes Unrechtsbewußtsein in der Öffentlichkeit zu erzeugen. Dies ist schon allein deshalb notwendig, um den harten Kern der Geschäftemacher von den Gelegenheitskopierern zu trennen, die mehr oder weniger aus Versehen in diesen Dunstkreis gelangt sind.«

Zum Schluß kam noch einmal die zeitliche Vorgehensweise bei der Verfolgung von Raubkopierern zur Sprache.

von Grafenreuth: »Man muß hierbei zwischen drei Kategorien unterscheiden. Zum einen sind dies professionelle Abmahnfirmen, wie der Fall R + S Software (vorgestellt in der Ausgabe 5/85, Seite 8 vom 64'er; Anm. der Red.), dann Abmahnungen von Computerläden und Mitbewerbern, die in der Regel nicht autorisiert sind, derartige Abmahnungen an einzelne Personen zu schicken. Mitbewerber haben keinen Anspruch auf Urheberrecht. Die dritte Kategorie bildet die Firma, welche das wirkliche Vertriebsrecht des Programmes hat. Hier muß man selbst entscheiden, ob man die Unterlassungserklärung unterschreibt oder nicht.«

Info: Bestimmte mathematische Verfahren und Lösungsansätze (Algorithmen) können nicht urheberrechtlich geschützt werden. So ist es beispielsweise erlaubt, die verschiedenen Sortieralgorithmen wie Quicksort, Shellsort oder Straight Insertion in eigenen Programmen zu verwenden. Wollen Sie ein Programm von einem Computer auf einen anderen übertragen, oder passen lediglich die Drucker- und Ausgaberoutinen Ihren eigenen Verhältnissen an, so erwerben Sie keine Rechte an diesen neuen Programmen. Sie sollten daher die Quellenangabe des ursprünglichen Autors mit in das »neue« Programm übernehmen. Dies entspricht auch den ethischen und moralischen Gepflogenheiten fairer Programmierer. Auf jeden Fall sollte im Falle einer Veröffentlichung oder kommerziellen Nutzung des eigenen Programms mit Fremdanteilen vorher die Zustimmung des Verlages oder des Autors eingeholt werden. Übersteigen die Veränderungen und Verbesserungen am ursprünglichen Programm mehr als 30 bis 40 Prozent, so kann dieses Programm als eigene Entwicklung angesehen werden. Doch auch hier sollte fairerweise wieder die Quellenangabe Bestandteil zumindest in Rem- und Print-Zeilen sein.

Die Diskussion um das Raubkopiererunwesen hat doch einige Aussagen der Rechtsexperten gebracht, die vorher wohl noch nie so deutlich ausgesprochen wurden.

Es ist erstaunlich, daß man das Kopieren für den rein privaten Gebrauch und auch die Weitergabe von Kopien an enge Freunde als nicht strafbar bezeichnete. Auch die Einstellung der meisten Verfahren gegen jugendliche Raubkopierer zeugt von dem Verantwortungsbewußtsein, das die Staatsanwaltschaft an den Tag legt.

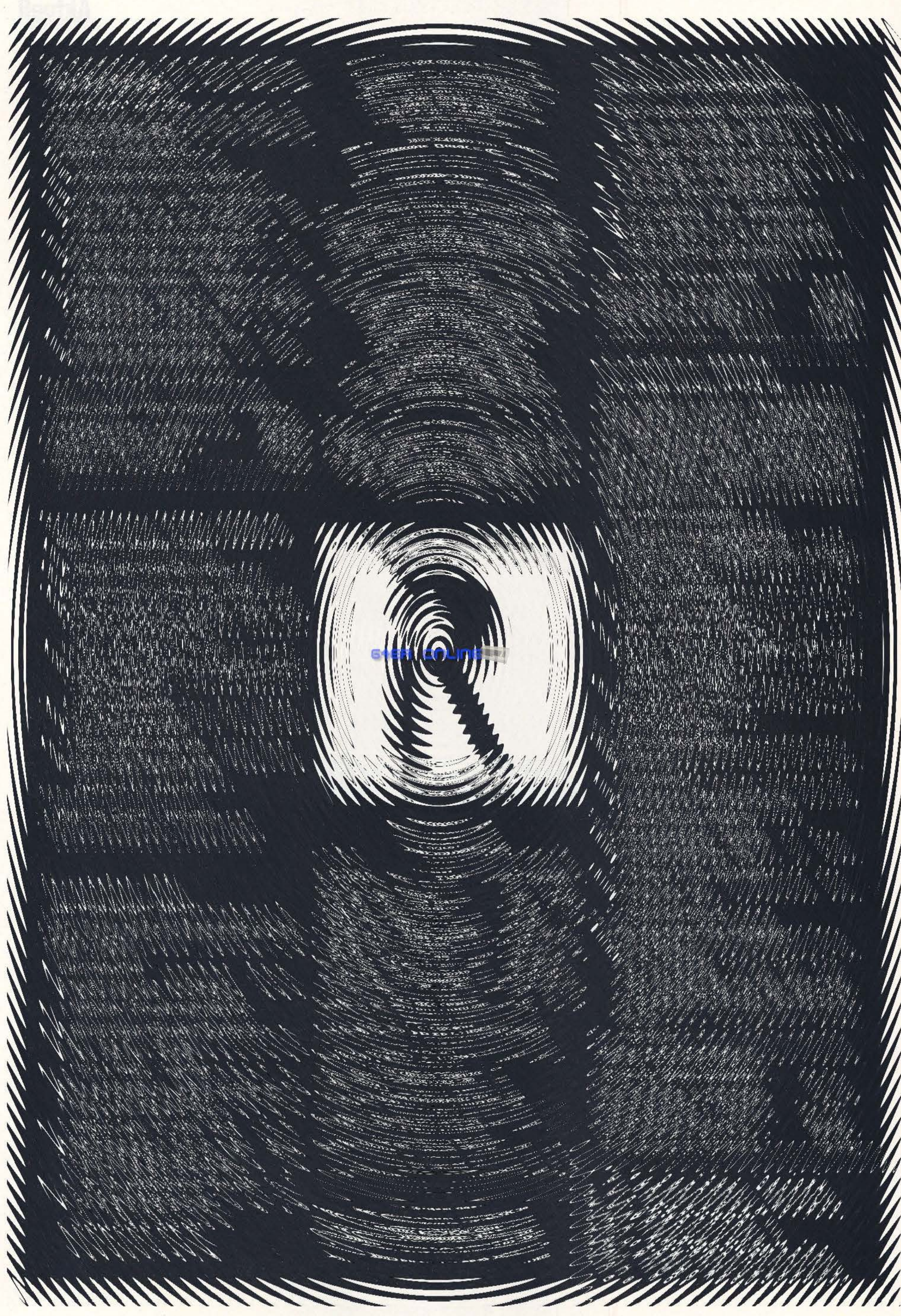
Dennoch bleiben einige Fragen und Ungereimtheiten. Viele Firmen erteilen beim Verkauf der Software dem Anwender lediglich das Nutzungsrecht an dem Produkt (das heißt, die Software bleibt weiterhin Eigentum des Herstellers). Das Kopieren des Originals wird ausdrücklich untersagt. Diese »Vertragsklauseln« gelten mit dem Öffnen der Verpackung von seiten des Käufers als akzeptiert.

Der praktikabelste Weg scheint wirklich eine Lösung nach dem Gema-Konzept zu sein. Dabei wird auf Leerdisketten ein bestimmter Betrag aufgeschlagen, der dann den Software-Herstellern zugute kommt. Dabei wird das Kopieren des Originals für private Zwecke erlaubt. Das Problem, daß auch die Personen, die Disketten ausschließlich für eigene Programmentwicklungen verwenden, für die Kopierer die Mehrkosten mittragen würden, ist sicherlich sekundär und hat auch bei den Tonbandkassetten zu keinen Schwierigkeiten geführt. Und seien wir doch einmal ehrlich: Wer hat nicht in seiner Sammlung mindestens eine Raubkopie? Gerade die Raubkopiererszene bei Software hat in Deutschland ein bisher nie gekanntes Ausmaß angenommen und läßt sich mit harten strafrechtlichen Maßnahmen nicht mehr eindämmen.

Die extreme Verbilligung der Software wäre ein anderer möglicher Weg, um das Kopieren von Programmen zu vermindern. Dadurch würde aber sicherlich die Qualität der Software erheblich leiden, denn es würde sich keine Software-Firma mehr bereit erklären, die hohen Entwicklungskosten für gute Programme zu tragen. Ebenso könnte der Fachhandel eine qualitativ hochwertige Beratung und Betreuung seiner Kunden nicht mehr gewährleisten.

Die 64'er Redaktion steht eindeutig auf seiten der Anwender, die von ihren Originalprogrammen Kopien ziehen, um sich vor einem eventuellen Datenverlust abzusichern. Auch die Weitergabe dieser Programme an gute Freunde (auch auf die Gefahr hin, dem Ostfriesenprinzip Vorschub zu leisten) kann nicht verurteilt werden. Vorgegangen werden muß allerdings drastisch gegen die skrupellosen Geschäftemacher, die sich auf Kosten der Firmen einen finanziellen Vorteil verschaffen wollen. Diese Leute sollten doch einmal selber gute Programme schreiben, ein Jahr ihres Lebens dafür aufwenden, um dann von »Kollegen« um die Früchte ihrer Arbeit gebracht zu werden. Wie würde wohl dann deren Reaktion aussehen? Wir treten für die freie Marktwirtschaft ein, und billigen jedem Unternehmen zu, Gewinn machen zu wollen und zu müssen, verurteilen allerdings unmotivierte Geschäftemacherei. Uns interessiert in diesem Zusammenhang die Meinung unserer Leser. Wie sehen Sie mögliche Ansatzpunkte, speziell die Gema-Lösung? Schreiben Sie uns Ihre Einstellung zu diesem Thema.

(aa)



Aktuelles von der C'85 in Köln

Die Kölner Messe zeigte es sehr deutlich: Das Zubehörangebot für Heimcomputer wird größer, die Software wird preiswerter

Fischertechnik für Computer

Bei Fischertechnik wird die Linie der »Computing-Baukästen« zügig ausgebaut. Ein allgemeiner Grundbaukasten ist bereits erhältlich. Zehn verschiedene, computergesteuerte Modelle können damit gebaut werden, von der Ampelanlage über eine Sortieranlage bis hin zu einem einfachen Roboter oder zu einem Grafiktablett.

Ab Herbst '85 erhältlich sind ein Trainings-Roboter mit drei freibeweglichen Achsen und ein Plotter-Scanner, der wahlweise zeichnen oder Bilder abtasten kann. Alle Computing-Baukästen lassen sich mit den »klassischen« Fischertechnik-Bausätzen kombinieren und ausbauen.

Um die Modelle betreiben zu können, braucht man außer einem Computer nur noch das entsprechende Interface, das zusammen mit Treibersoftware für verschiedene Computer, unter anderem natürlich auch für den C 64, ebenfalls lieferbar ist.

Info: Fischertechnik-Bausätze der Computing-Serie gibt's im Spielwaren-Fachhandel und in Kaufhäusern. Bezugsquellennachweis und nähere Information von Fischer-Werke, Artur-Fischer GmbH & Co. KG, Weinhalde 14-18, 7244 Tumlingen/Waldachtal.

DFÜ und Btx für 398 Mark

Umschaltbar von 300 auf 75/1200 beziehungsweise auf 1200/75 Baud ist der neue Akustikkoppler AK 2000 S. Dieser Koppler besitzt für beide Betriebsarten die FTZ-Nummer 18.13.1997.00. Auf den AK 2000 S wird eine Garantie von sechs Monaten gewährt. Mit 398 Mark ist der Preis als sehr günstig zu bezeichnen.

Info: Jochen Gerhardt & Bettina van Megern GbR, Höhenstr. 74 B, 4000 Düsseldorf 1.

Software Express GmbH, Hugo Viehoffstr. 84, 4000 Düsseldorf 30

Neues Btx-Modul für C 64

Das Modul »Btx 64« ermöglicht den preiswerten Einsatz des C 64 als Btx-Tastatur und den Abruf von Telesoftware. Neben dem Speichern von Btx-Seiten können diese Offline verändert und auf dem Drucker ausgegeben werden. Wiederkehrende Tastenbetätigungen können gespeichert und automatisch abgerufen werden. »Btx 64« kostet 248 Mark. Um dieses System für alle Btx-Decoder anzubieten zu können, kommen noch 29,80 Mark für einen entsprechenden Kabelsatz hinzu. Aktuelle Programme und Handbucherweiterungen für »Btx 64« sind über Btx abrufbar.

Info: Astech Computer GmbH, Am Wall 183, 2800 Bremen 1, Tel. 0421/324058

Software von Sybex

Der bisher durch eine Vielzahl von Computer-Büchern bekannte Sybex-Verlag bietet jetzt auch Software an. StarTexter ist ein komfortables Textverarbeitungs-Programm für den C 64, das neben deutschen Umlauten, automatischer Formatierung und Trennhilfe auch den Entwurf eigener Zeichensätze ermöglicht. Daneben kann StarTexter auch noch rechnen und läßt sogar Basic-Programmierung zu. StarTexter besteht aus einem Textverarbeitungskurs in Buchform und einer Programmdiskette. Der Preis ist ebenfalls an den C 64 angepaßt: StarTexter kostet 64 Mark.

Info: Sybex Verlag GmbH, Postfach 300961, 4000 Düsseldorf

Neuer Brother Typenrad-Drucker

Der neue Brother HR-10C druckt bidirektional mit bis zu drei Durchschlägen bei einer Geschwindigkeit von 12 Zeichen pro Sekunde. Das Schriftbild ist wie von Typenradmaschinen gewohnt sehr gut und durch die Möglichkeit des Typenradwechsels auch sehr flexibel. Der HR-10C ist direkt anschlußfertig an den C 64 und wird unter 900 Mark kosten.

Info: Brother International GmbH, Im Rosengarten 14, 6368 Bad Vilbel

Neue Bezugsadresse für Oxford Pascal

Der Oxford-Pascal-Compiler für den C 64 (Test in Ausgabe 12/84) wird nach dem Konkurs der bisherigen Vertriebsfirma CPL in Deutschland weitervertrieben durch die Firma Gepo Soft. Gepo Soft betreut auch alle Oxford-Pascal-Anwender im deutschsprachigen Raum bei Problemen oder Fragen. Die bereits angekündigte Umtauschaktion für die Version 1.1 des Compilers wird ebenfalls von Gepo Soft durchgeführt. Gegen Einsendung der Originaldiskette und 60 Mark erhält man eine Diskette mit der verbesserten Version. Backup-Disketten sind ebenfalls erhältlich.

Info: Gepo Soft, Gubener Straße 23, 4650 Gelsenkirchen.

Neue C 64-Software

Neue C 64-Anwenderprogramme und Spiele bietet Data Media an. Neben den »klassischen« Anwendungen wie Textverarbeitung, Lager- und Adreßverwaltung sind unter anderem Reisekostenabrechnung, Faktura, Terminplaner und diverse Archivprogramme erhältlich. Auf dem Spielesektor gibt es neben Remakes von altbekannten Spielen für den C 64 wie Skramble, Galaxy oder Moonbuggy auch neue, aktuelle Spiele wie Slap Shot (Eishockey), Captain Starlight oder Twilight Zone. Der Preis liegt für die Kassettenversionen zwischen 27 und 34 Mark bei den Spielen und zwischen 57 und 177 Mark bei den Anwenderprogrammen. Auch Programme für VC 20, C 16/116 und Plus 4 sind erhältlich.

Info: Data Media GmbH, Ruhrallee 55, 4600 Dortmund.

Neue Spiele zu Tiefstpreisen

Jede Menge Spielesoftware für den C 64, aber auch für den C 16/116, Plus 4 und VC 20 bietet Mastertronic an. Dabei handelt es sich nicht etwa um neuaufgelegte Pac-Man- und Space-Invader-Versionen, sondern alle Spiele sind brandneu auf dem deutschen Markt. Selbst bei den VC 20 - Spielen handelt es sich durchweg um neue, in Deutschland noch nicht veröffentlichte Programme. Alle Programme enthalten eine kurze deutsche Anleitung und sind mit einem sehr guten Kopierschutz, nämlich mit einem Preis von sage und schreibe nur 11,95 Mark ausgestattet.

Info: Mastertronic-Programme gibt es im Computer- und Spielefachhandel und in Kaufhäusern sowie direkt bei Mastertronic, Kellas Computer Vertrieb, Riga-Ring 6, 4770 Soest/Westfalen.

Dynamics erweitert Zubehör-Angebot

Unter der Bezeichnung »CMK-49« stellte Dynamics in Köln ein Musik-Keyboard zum direkten Anschluß an den C 64 vor. Das CMK-49 soll einschließlich Handbuch und komfortabler Software auf Diskette oder Kassette 398 Mark kosten.

Wer nicht ganz so hohe Anforderungen stellt, kann für 98 Mark eine Musik-Keyboard-Auflage CMK-25 erhalten. Dieses Modell wird einfach auf die C 64-Tastatur aufgelegt. Der Anwender betätigt damit indirekt die C 64-Tastatur. Das CMK-25 wird ebenfalls mit Software und Handbuch ausgeliefert.

Des weiteren wurde bei Dynamics erstmalig für den C 64 eine »Maus« vorgestellt. Der Anschluß erfolgt über die Joystick-Ports. Die Dynamics-Maus arbeitet nach dem Rollkugel-Prinzip und besitzt drei zusätzliche Schalter. Damit wird komfortable Menüsteuerung auch auf dem C 64 möglich. Der Preis von wahrscheinlich um 130 Mark für die Maus versteht sich inklusive Handbuch und komfortablen Malprogramm mit ausgefeilter Menüsteuerung als Anwendung.

Info: Dynamics Marketing GmbH, Postfach 112005, 2000 Hamburg 1.

Wohin mit dem Computer?

Für alle diejenigen, die das Computerhobby etwas ernsthafter betreiben und infolgedessen einige Unterbringungsprobleme bekommen haben, gibt es jetzt ideenreich konstruierte Spezial-Computermöbel (»RCR«, Rosita Computer Racks). Da bringt man nicht nur Computer, Floppy-Laufwerk und Monitor einfach und platzsparend unter, sondern, je nach Modell, bleibt auch noch genug Platz für Drucker, Diskettenbox und 64'er-Magazine.

Info: Information und Bezugsquellennachweis bei Rosita Tonmöbel, Theo Schmitz GmbH & Co. KG, Postfach 6320, 4790 Paderborn — Schloß Neuhaus

C-Compiler von Data Becker

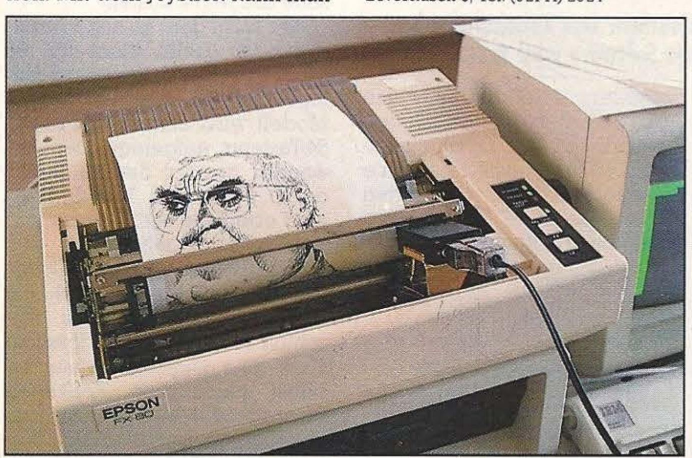
Als erstes Softwarehaus bietet Data Becker für den C 64 einen Compiler für die Programmiersprache C an. C vereint Maschinennähe und komfortable Programmierung in fast idealer Weise. Ein Test dieses Compilers folgt in einer der nächsten Ausgaben. (ev)

Info: Data Becker, Merowingerstr. 30, 4000 Düsseldorf

Scanner

Scan 64 Newtronics ist ein Digitalisiergerät, das auf den Druckkopf des FX-80 aufgesetzt wird (Einbausätze für andere Drucker sind in Vorbereitung). Die Steuersoftware ermöglicht das Einlesen von Bildern, die um ein vielfaches die Größe des Bildschirmes übersteigen können. Mit dem Joystick kann man das gescannte Grafikbild Stück für Stück durch Scrolling nachbearbeiten. Dazu stehen Funktionen eines normalen Zeichenprogramms wie zum Beispiel Draw, Circle und Zoom zur Verfügung. Die Hardcopyroutine druckt die gescannte Vorlage in zwei verschiedenen Größen.

Info: Rolf Rocke Computer, Auestr. 1, 5090 Leverkusen 3, Tel. (02171) 2624



Neu und doch nicht neu

In der Ausgabe 3/85 auf Seite 14 stellten wir ein »neues« 1541-Laufwerk vor. Es handelte sich dabei aber lediglich um eine »Übergangslösung«, wie uns Commodore mitteilte. Dieses Diskettenlaufwerk, erkennbar an dem Knebelverschluß auf der Frontseite, soll vollkommen identisch mit dem 1541-Laufwerk mit Kippverschluß sein. Commodore hatte vor einem Jahr aufgrund der Nachfrage mit einem Lieferengpaß zu kämpfen. Deshalb wurde ein gesonderter Auftrag an eine japanische Firma vergeben, 10000 1541-Floppies mit Mitsumi-Laufwerken zu produzieren. In der »normalen« 1541-Floppy werden in der Regel in einem Verhältnis von 50:50 sowohl Alps- als auch Newtronics-Laufwerke eingebaut. Die Qualitätsmerkmale beider Laufwerke seien vollkommen identisch. Wir ließen uns durch den optisch veränderten Aufbau der in Japan produzierten Laufwerke dazu verleiten, anzunehmen, daß es sich dabei um eine verbesserte Diskettenstation handle. Dem sei nicht so, sagte Commodore. Schlußfolgerung: Die 1541 hat ihr Gesicht nicht verändert. Es gibt nach wie vor nur eine einheitliche Station mit zwei verschiedenen Laufwerks-Lieferanten. Die »Knebelversion« ist eine auf 10000 Stück begrenzte Ausnahme. Wieder nichts gewesen, könnte man da sagen.

(aa)

Kuriositätenecke

— Nolan Bushnell, Gründer von Atari, hat nach mehreren fehlgeschlagenen Versuchen, neue Firmen aufzubauen (Pizza Time Theater, Sente, Androbot) wieder einen neuen Anlauf gestartet. Die neue Firma »Axlon« hat eine besondere Art von Robotern anzubieten: Die »Petsters« sollen Haustiere ersetzen. Ein »Catster« beispielsweise miaut, wenn er in die Ecke gesetzt wird, murrt behaglich, wenn er gestreichelt wird, aber kratzt nicht und frißt nur Batterien. »Dogster« und »Catster« können bis zu 250 vorprogrammierte Funktionen wahrnehmen. Bushnell: »Wir haben es geschafft, etwa 80 Prozent der Funktionen eines Haustieres »nachzubilden«. So ein Haustier kostet übrigens 100 US-Dollar. Miau!

— Daß Bücher zu Computerspielen gemacht werden, ist nichts Neues. Das Umbauen eines Computerspiels zum Buch ist relativ neu. Nachdem Alan Dean Fosters Buch »Shadowkeep« (ein zu empfehlender Fantasie-Roman) nach dem gleichnamigen Spiel von Telarium entstand, sind nun auch Zork-Bücher erhältlich. Die allerdings sind keine gewöhnlichen Romane, sondern Adventures ohne Computer. Man geht dabei ähnlich vor wie bei Computerabenteuern:

Nachdem man gelesen hat, wo man ist und was passiert, wählt man einen Befehl aus. Entsprechend dieses Befehls wird man dann auf die richtige Seite verwiesen, wo das Spiel dann seine Fortsetzung findet. Die Zork-Bücher (vier Stück gibt es) sind übrigens von Steve Meretzky geschrieben, der auch Autor der Infocom-Adventures Planetfall und Sorcerer ist. Der Preis für ein solches Buch: 2 Dollar. So billig müßte Software sein...

— Wer wissen will, woher wir immer die Informationen für unsere Kuriositätenecke haben: Fast alle Zeitschriften der internationalen Presse werden bis auf den letzten Buchstaben durchwühlt, ob amerikanisch, deutsch, englisch oder sonst woher. Ab und zu bietet sogar der amerikanische Playboy interessante Neuigkeiten aus dem Computer-Business. Und ganz nebenbei gibt es noch Informationen aus der »Szene« — Händler, Hacker und Spiele-Enthusiasten — wobei der Unterschied zwischen diesen Kategorien oft gleich Null ist.

— Die Superhelden-Welle rollt weiter. Nachdem Adventure International gemeinsam mit Commodore die Marvel-Comic-Helden lizenzierten (»The Hulk«, »Spiderman«) beginnt First Star Software, auf dieser Welle mitzureiten. Der Vertrag mit DC-Comics beinhaltet die Rechte für »Superman«, »Wonderwoman« und »Darkseid«-Comics.

— Roe R. Adams III, Redakteur der führenden amerikansichen Computerspiel-Zeitschrift (Computer Entertainment) und Mitautor der Fantasyspiel-Klassiker Wizardry IV und Ultima IV, hat mehr Spiele zu Hause stehen, als er benötigt. Aus diesem Grund verschenkte er sage und schreibe 1000 Spiele an die Gutman-Bibliothek in Harvard (wo man sie ausleihen kann). Jetzt hat er »nur« noch 6000 Spiele.

— Die ersten 60000 Exemplare des computerisierten Comics »Shatter« waren innerhalb von vier Tagen ausverkauft. Die Grafiken wurden mit einer Maus auf dem Macintosh gemalt. Die Comicautoren führen den großen Erfolg auf Computer-Enthusiasten zurück.

— Aus einem amerikanischen Computermagazin: Das Programm »Mind Prober«, mit dessen Hilfe man die Psyche anderer (angeblich) analysieren kann, wurde mit Daten über Ronald Reagan gefüttert. Mind Prober erstellte diesen Output:

- Er ist ein »Crowd-Pleaser«, also jemand, der immer tut, was nötig ist, um die Massen zu erfreuen.
- Man sollte sich vor seinen Versprechen in Acht nehmen. Er gibt sie leicht und hält sich nicht immer dran.
- Diese Person würde es lieben, etwas zu tun, was ihn in den Mittelpunkt des Interesses stellt. Also Schauspieler oder Politiker.

Hat die Software hier nicht ein bißchen Wahrheit eingefangen?

— Commodore kann übrigens Diskettenlaufwerke schon selbst reparieren. Da aber der VC 20 auch bei den Commodore-Technikern »Out« ist, werden die eingelieferten Geräte gesammelt und zehntausendstückweise nach Japan gesandt. Commodore Japan repariert die Geräte und schickt sie dann zurück. Und da soll sich noch einer wundern, warum er so lange auf seine Geräte warten muß. Ein bißchen kurios ist diese Methode schon...

— Thomas Tempelmann, Autor des weltberühmten »FCopy«, verdient nun endlich doch ein bißchen Geld an seinem Programm. Nachdem er nur 70 Originale verkaufen konnte (und Tausende C 64-Besitzer das Programm haben), klagte er sein Leid während der Commodore-Messe (CFA 1984) dem Commodore-Guru Jim Butterfield. Der hat daraufhin einen ganzseitigen Artikel im Magazin des weltgrößten Commodore-Clubs (TPUG) über Thomas Tempelmann veröffentlicht. Daraufhin waren natürlich viele Amerikaner »betroffen« von den Ausmaßen des Raubkopierens (ein Autor schreibt ein epochemachendes Werk und verdient keinen Pfennig daran), bis es zu einem Aufruf kam, Thomas ein bißchen Geld zu spenden und ihn so für seine Mühe zu entlohnen. Und an den laufend eintrudelnden 5-Dollar-Scheinen verdiente er bislang etwa 4000 bis 5000 Mark. Thomas' Kommentar: Aus Deutschland würde ich trotz Spendenaufruf nicht mal 5 Mark erhalten.



64er online



Erfahrungen gesucht

Wer hat bereits Erfahrungen mit der »Megafloppy« SFD 1001 gesammelt, insbesondere im Hinblick auf die Zusammenarbeit mit C 64, 1541-Floppy und Speeddos? Gibt es ein Programm, welches die Daten einer mit der 1541 bespielten Disk sicher auf SFD 1001-Disketten überträgt?

Wer hat Erfahrungen mit der Schreibmaschine Queen Data Executive 80 gesammelt?

Braucht man ein spezielles Interface, um sie am C 64 zu betreiben? Ist ein Betrieb mit Vizawrite möglich?

Bernhard Abel

Midi-Software gesucht

Wer hat Erfahrung mit Midi-Software? Wo bekommt man ein Midi-Interface zum Anschluß des C 64 an den Korg Poly 61M?

Mathias Heck

Fragen Sie doch

Selbst bei sorgfältiger Lektüre von Handbüchern und Programmbeschreibungen bleiben beim Anwender immer wieder Fragen offen. Viel mehr Fragen ergeben sich bei Computer-Interessenten, die noch keine festen Kontakte zu Händlern, Herstellern oder Computerclubs haben. Sie können der Redaktion Ihre Fragen schreiben oder Probleme schildern (am einfachsten auf der Karte »Lesermeinung«). Wir veranlassen, daß sie von einem Fachmann beantwortet werden. Allgemein interessierende Fragen und Antworten werden veröffentlicht, die übrigen brieflich beantwortet.

Textverarbeitung mit Formeln?

Ich besitze einen Seikosha GP-550-Drucker mit VC-Interface und suche ein Textprogramm, mit dem ich mathematische Sonderzeichen und hoch-/tiefgestellte Indizes ausdrukken kann.

Wer hat eine Hardcopy-Routine für den GP-550, die möglichst auch mit Simons Basic zusammenarbeitet?

Johann Bierschneider

Drucker und 80-Zeichen-Karte?

Ich besitze einen Commodore 64, welcher durch ein geändertes Betriebssystem in der Lage ist, den User-Port als Centronics-Schnittstelle zu nutzen. Damit betreibe ich eine Typenradschreibmaschine Olympia Compact 2 und eine 80-Zeichen-Karte der Firma Roos.

Wenn nun die 80-Zeichen-Karte in Betrieb ist, läßt sich der Drucker nur noch über das von der Firma Roos mitgelieferte Textverarbeitungsprogramm ansprechen. Alle Versuche, den Drucker von Basic aus mit den üblichen Befehlen anzusteuern, schlugen fehl. Mehrere Anrufe bei der Firma Roos und der Herstellerfirma Zero in Holland brachten außer hohen Telefonkosten kein Ergebnis.

Wer kann mir helfen (Soft- oder Hardware), den Drucker zusammen mit der 80-Zeichen-Karte zu betreiben?

Michael Birkners

Orgel am C 64?

Wer kann mir Tips geben, wie ich meine Orgeltastatur (49 Tasten) an den C 64 anschließen kann?

Thorsten Markus

CPU sockeln?

Ich habe vor, die in Ausgabe 1/85 getestete 64 KByte-Erweiterung für den VC 20 zu kaufen. Doch leider ist bei meinem VC 20 die CPU nicht gesockelt. Kann ich die Erweiterung dennoch verwenden oder muß ich die CPU erst sockeln lassen?

Denis Crede

Ein 40poliger Sockel guter Qualität kostet etwa vier bis fünf Mark; wenn Sie die CPU beim Händler sockeln lassen, zahlen Sie zusätzlich noch einiges an Arbeitslohn. Zudem besteht die Gefahr, daß der Prozessor bei aller Vorsicht den Auslötvorgang nicht überlebt. Unser Vorschlag ist daher: Besorgen Sie sich im Elektronikladen eine neue 6502 CPU (kostet 12 bis 15 Mark) und stecken Sie diese in den dafür vorgesehenen Sockel der 64 KByte-Erweiterung. Mit einer kleinen Zange kneifen Sie jetzt die Pins der Original-CPU ab, werfen diese weg und löten die stehengebliebenen Pins aus. Die Erweiterungsplatine stekken Sie anstelle der soeben ausgebauten CPU auf die Rechnerplatine und löten sie vorsichtig ein. Das ist die wohl schnellste und preiswerteste Lösung Ihres Problems.

Zwei Datasetten am Commodore 64

Auf der internen Platine der Datasette befindet sich eine Steckerleiste, auf die der Datasettenstecker haargenau paßt. Läßt sich darüber eine zweite Datasette anschließen? Ausgabe 3/85

Uwe Bilo

Die Pin-Belegung des Platinensteckers in der 1530/C2N ist folgende:

1 +5V
2 Read Data
3 Write Data
4 Masse (GND)
5 Head hot
6 Head return

Während an den Pins 2 und 3 die gleichen Signale wie am computerseitigen Stecker liegen, sind die Pins 5 und 6 direkt mit dem Tonkopf verbunden. Diese Anschlüsse sollte man also mit Vorsicht behandeln. Sie sind vor allem interessant für Elektroniker, die ein Oszilloskop zur Hand haben und damit einen verstellten Tonkopf wieder justieren können.

Es ist möglich, eine zweite Datasette anzuschließen, wenn man einfach die entsprechenden Anschlüsse der Datasetten parallel auf den Kassettenport legt. Da jedoch weder die Datasette von sich aus über eine bestimmte Geräteadresse verfügt, noch der C 64 zwei verschiedene Geräteadressen für Datasetten bereitstellt, wird man dabei allerdings auf folgende Probleme stoßen:

1. Beim Laden/Speichern mit einer Datasette läuft automatisch auch der Motor der anderen mit.
2. Der Computer kann nicht unterscheiden, an welcher Datasette eine Taste gedrückt ist. Die Entscheidung, welche Datasette nun angesprochen werden soll, liegt also immer beim Benutzer, der Computer spricht stets beide gleichzeitig an.
3. Der C 64 hat natürlich trotz allem nur einen Kassettenpuffer, es ist also beispielsweise nicht möglich, auf der einen Datasette eine Datei zum Lesen zu öffnen und auf der anderen eine Datei zu schreiben.

Um Komplikationen zu vermeiden, wäre es nötig, die Motorsteuerung beider Datasetten über den User-Port vorzunehmen und auch das Betriebssystem per Maschinenprogramm an die geänderten Verhältnisse anzupassen. Der Anschluß einer zweiten Datasette ist also nicht so einfach, wie es auf den ersten Blick aussieht.

Bertram Dunskus

Wollen Sie antworten?

Wir veröffentlichen auf dieser Seite auch Fragen, die sich nicht ohne weiteres anhand eines guten Archivs oder aufgrund der Sachkunde eines Herstellers beziehungsweise Programmierers beantworten lassen. Das ist vor allem der Fall, wenn es um bestimmte Erfahrungen geht oder um die Suche nach speziellen Programmen. Wenn Sie eine Antwort auf eine hier veröffentlichte Frage wissen — oder eine andere, bessere Antwort als die hier gelesene, dann schreiben Sie uns. Antworten publizieren wir in einer der nächsten Ausgaben. Bei Bedarf stellen wir auch den Kontakt zwischen Lesern her.

1526-Betriebssystem entschleiert

Wie kann ich feststellen, ob mein 1526-Drucker mit dem alten oder mit dem neuen Betriebssystem ausgerüstet ist? Ausgabe 3/85, 5/85

Günter Reuter

Um zu erfahren, welches Betriebssystem Ihr 1526 hat, müssen Sie nur den Druckerselbsttest durchführen (Paper-Advance-Taste beim Einschalten gedrückt halten). Der Drucker gibt dann in der Kopfzeile die Version aus.

Um endlich mit der Vorstellung aufzuräumen, es gäbe zwei Betriebssysteme: Es gibt deren fünf!

REV 05 Diese Version ist gekennzeichnet durch einen Fehler in der Ausgaberoutine: Bei Zeilen, in denen nur Leerzeichen vorkommen, erscheinen manchmal wirre Zeichen auf dem Papier. Diese Version ist zwar grafikfähig, durch den obengenannten Fehler sehen die Grafiken jedoch nicht sehr schön aus.

REV.07 und **REV-07** Diese Versionen zeichnen sich durch absolute Grafikunfähigkeit aus. Beim Versuch, ein umdefiniertes Zeichen zu benutzen, wird der Zeilenvorschub blockiert.

REV:07 Dieses Betriebssystem funktioniert fast einwandfrei. Hochauflösende Grafik ist möglich, allerdings werden einige seltene Sonderzeichen in Basic-Listings falsch gedruckt.

REV 1.0 Bis auf einen stellenweise etwas abgeänderten Zeichensatz ist diese Version mit »REV:07« identisch.

Zusammenfassend läßt sich sagen, daß nur die Versionen »REV 1.0« und »REV:07« zufriedenstellend funktionieren. Eine Umrüstung auf andere Versionen ist durch das Auswechseln des EPROMs im Drucker möglich. Entsprechende Angebote finden sich von Zeit zu Zeit im Anzeigenteil.

Rainer Wiesenfarth

Leser fragen — Willi Brechtl antwortet

Hallo liebe Leser, hier bin ich wieder, um Eure Fragen zu beantworten.

Ich werde mich hauptsächlich um Leserbriefe kümmern, die nicht in das sachliche Einerlei des Leserforums passen. Zum Beispiel Fragen, die sich aus dem einen oder anderen Grund nur ganz subjektiv beantworten lassen. Oft genug tauchen auch Probleme auf, die sich nicht mit einem kurzen Antwortsatz abhandeln lassen. Und wenn selbst eine längere Antwort im Rahmen des Leserforums nicht mehr ausreichen würde, dann ist das ganz klar ein Fall für Willi Brechtl.

Also: Wenn Sie als Anfänger Probleme mit Computer, Software oder Handbuch haben, dann wenden Sie sich in Zukunft doch einfach vertrauensvoll direkt an mich.

C 16: Bilder auf Datasette?

Wer kann mir sagen, wie man beim C 16 fertige Bilder außer mit Shapes auf die Datasette speichert?

Hardeen Hornburg

Speichererweiterung umschaltbar

Bei meiner schaltbaren 32/27 KByte-Erweiterung für den VC 20 der Firma Klaus Jeschke ist der Speicherbereich ab $4000 nicht umschaltbar. Ausgabe 3/85

Frank Heynig

Lösen Sie die Schraube auf der Unterseite des Moduls. Nehmen Sie die Platine aus dem Gehäuse und drehen Sie sie so, daß die Lötseite nach oben zeigt. Wenn Sie sich nun die Lötstellen des DIL-Schalters ansehen, werden Sie feststellen, daß die Kontakte des $4000-Schalters überbrückt sind. Dadurch ist der $4000-Bereich ständig eingeschaltet.

Sie brauchen also nur die Lötbrücke wegkratzen, um den $4000-Bereich ebenso wie die anderen Speicherbereiche schalten zu können. Meine Speichererweiterung funktioniert seit diesem unkomplizierten Eingriff einwandfrei. Sie sollten aber dennoch daran denken, daß mit dem Öffnen des Gehäuses der Garantieschutz verlorengeht.

Tobias Nicol

VC 20-Simulator gesucht

Ich interessiere mich sehr für einen VC 20-Simulator auf dem C 64. Existiert so ein Programm überhaupt?

Hellmut Korndörfer

Nein, so etwas gibt es bestimmt nicht. Der VC 20 ist sowieso im Basic identisch zum C 64, und ein Simulator kann in der Regel nur das Basic simulieren, nicht aber die völlig unterschiedliche Hardware.

Wie ist das mit dem Copyright?

Woher bekommt man das Copyright für seine Programme? Bringt es irgendwelche Vorteile, wenn man es hat?

Hellmut Körndorfer

Das Copyright besitzt man automatisch für jedes Programm, das man selbst geschrieben hat. Ein Copyrightvermerk in einem Programm (oder in einem Buch) dient nur dazu, anderen Personen anzuzeigen, bei wem die Rechte für das Produkt liegen, und daß man das Programm (oder Buch) nicht einfach kopieren und weiterverkaufen darf.

Leichter und besser als Basic?

Welche zweite Programmiersprache, die leichter zu erlernen und leistungsstärker als Basic ist, würden Sie mir empfehlen?

Markus Hautmann

Eine derartige Computersprache werden Sie kaum finden. Es ist ähnlich wie mit der Frage nach einem Sportwagen, der größer, schneller und besser als ein Porsche ist, aber auf jeden Fall viel weniger kosten soll. Beide Faktoren, Leistungsstärke und leichte Erlernbarkeit, schließen sich in der Regel aus. An sich ist Basic die am leichtesten zu lernende Programmiersprache. Einfache Grafikprogrammierung erreichen Sie mit Comal (sehr empfehlenswert, da auf Basic aufbauend) oder mit Logo. Mit Pascal läßt sich besonders »elegant« programmieren, es ist allerdings nicht ganz einfach zu lernen. Wenn es Ihnen auf Geschwindigkeit ankommt, sollten Sie sich vielleicht einmal mit Assembler oder Forth beschäftigen. Oftmals tut allerdings auch schon eine Basic-Erweiterung oder ein Compiler gute Dienste. Bei der Auswahl einer Programmiersprache kommt es halt immer darauf an, was man programmieren möchte. Dazu müßten Sie sich zuerst mal darüber klar werden, was Ihnen in Basic fehlt.

Assembler und andere Mysterien

Ich habe drei Fragen an Sie:
1. Was ist ein Assembler?
2. Was ist ein Disassembler?
3. Was ist ein Maschinensprachemonitor?
Was kann man damit machen, und wie funktionieren sie?

Michael Pitz

Wie Sie vielleicht wissen, versteht der im C 64 arbeitende 6510-Prozessor Basic nicht direkt, sondern nur eine viel einfacher (für die Maschine) aufgebaute »Sprache«, eben die sogenannte Maschinensprache. Sie können mit Ihrem Computer nur deshalb in Basic arbeiten, weil der C 64 ein Programm, selbst in dieser Maschinensprache geschrieben, fest eingespeichert hat, das einen »Basic-Prozessor« darstellt (genauso, wie Sie in Basic ein Programm schreiben können, das zum Beispiel eine Dateiverwaltung darstellt).

Aber natürlich können Sie Ihren C 64 auch in Maschinensprache direkt programmieren — das ist ja quasi seine Muttersprache. Allerdings geht das von Basic aus nicht ganz so einfach, genauso, wie Sie aus einem laufenden Dateiverwaltungsprogramm heraus schlecht in Basic programmieren können. Daher gibt es spezielle Hilfsprogramme, die das ermöglichen. In der einfachsten Form handelt es sich dabei um Maschinensprachemonitore, mit denen man — vereinfacht gesagt — kleine Maschinenprogramme gleich in der dem Prozessor geläufigen Zahlenform eingeben kann.

Dieses Programmieren in Zahlen ist zwar für die Maschine sehr angenehm, aber weniger für den Menschen. Daher wurden zu jedem Maschinenbefehl sogenannte »Mnemonics«, also leicht merkbare Abkürzung für die Wirkungsweise des Befehls, geschaffen. Um diese Mnemonics wiederum in das der Maschine verständliche Zahlenformat zu übersetzen, gibt es wiederum andere Programme — die Assembler.

Ein Disassembler wiederum tut das Gegenteil vom Assembler: Mit ihm können fertige Maschinenprogramme wieder in die Assembler-Mnemonics zurückübersetzt werden.

C 64 im VC 20-Gehäuse?

Kann man die Platine des C 64 in den VC 20 einbauen?
Wenn ja, wo bekommt man eine solche?

Hans-Willi Raedt

Prinzipiell ist dies ohne weiteres möglich, aber das bringt keinen finanziellen Vorteil ein, denn die Platine als Ersatzteil kostet mehr als der ganze Computer.

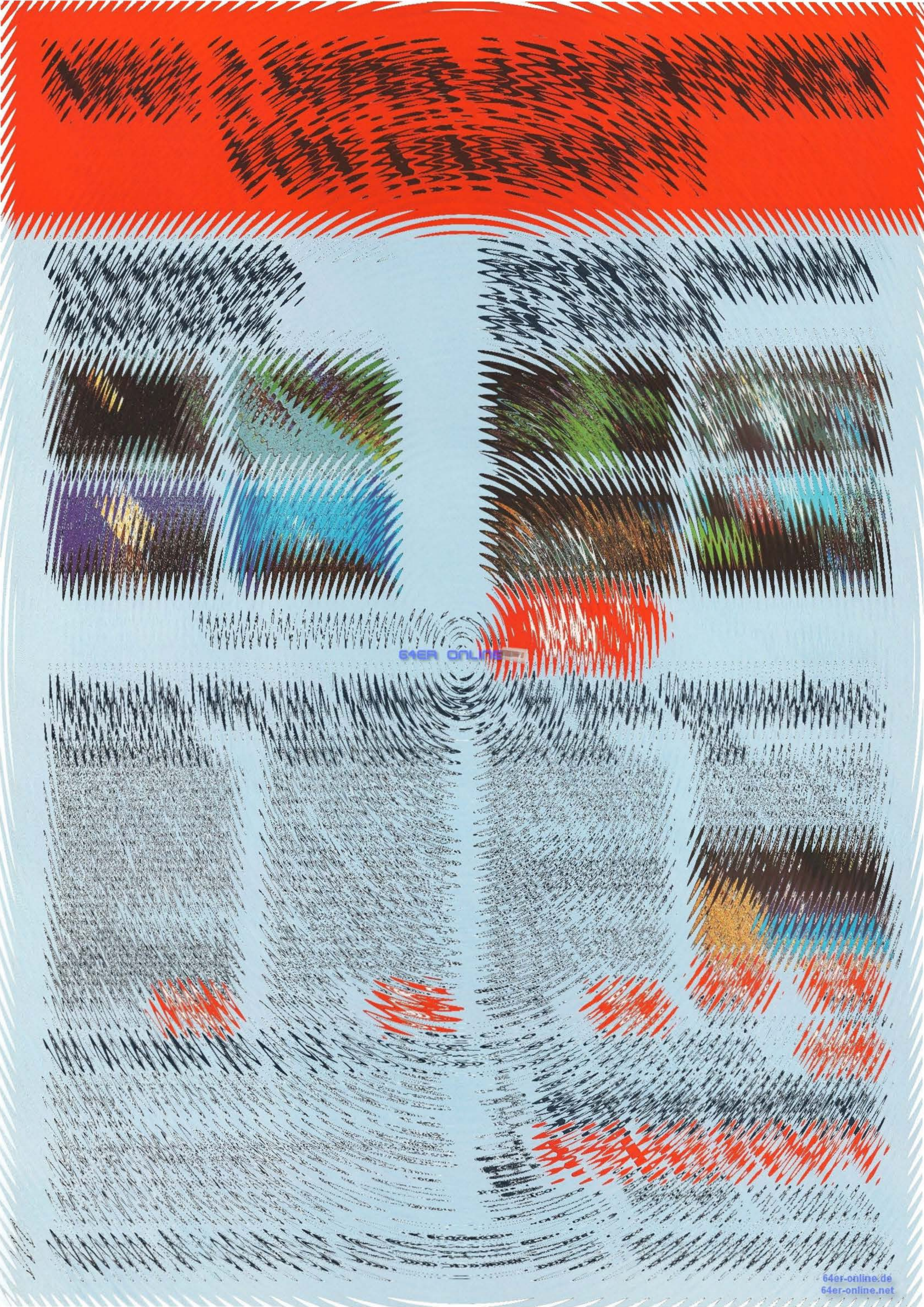
Mein Vorschlag daher: Sehen Sie sich doch einmal in einem Kaufhaus um! Dort bekommen Sie eine fertig ins Gehäuse verpackte Platine inklusive Tastatur, Netzteil und Handbuch. Auf dem Gebrauchtcomputermarkt werden des öfteren auch sehr günstige C 64 angeboten.

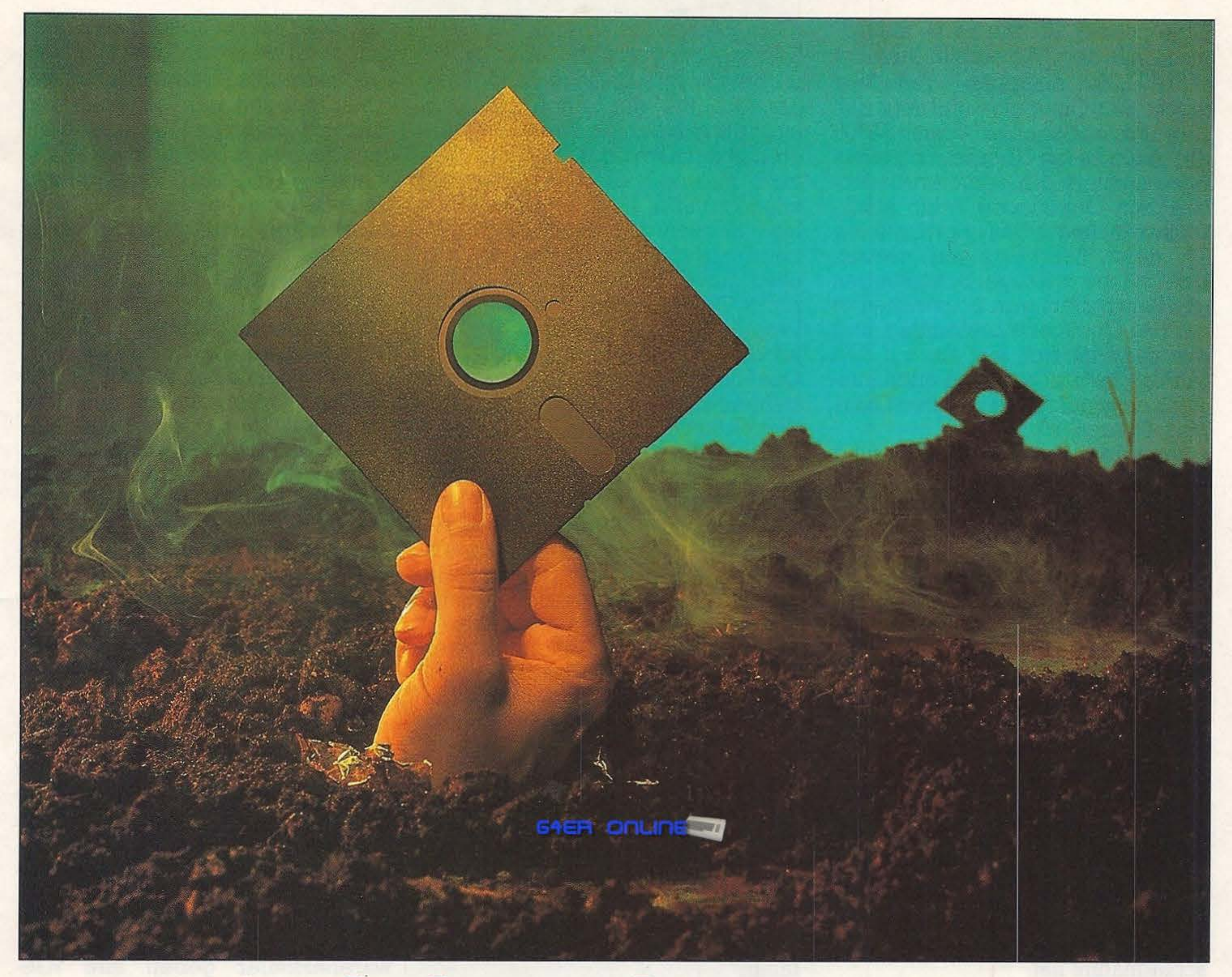
Ghostbusters-Probleme

Die Original-Kassette von »Ghostbusters« läuft auf meinem C 64 nicht. Bei einem Freund gibt es dagegen mit der gleichen Kassette keine Probleme.

Stefan Bardos

Wer hat ähnliche Probleme mit seinem C 64?





Das Urheberrechtsgesetz und Gedanken zu seiner Anwendung

Das Urheberrechtsgesetz soll die Urheber von Werken der Literatur, Kunst und Wissenschaft schützen (§1 UrhG). Allerdings steht dort nichts über Software.

Die geschützten Werke sind laut §2 Abs. 1:
1. Sprachwerke
2. Werke der Musik
3. Pantomimistische und Tanzkunstwerke
4. Werke der bildenden Künste (Baukunst, Kunstwerke)
5. Lichtbildwerke
6. Filmwerke
7. Darstellungen wissenschaftlicher und technischer Art

Programme, also »Computerwerke«, sind noch nicht besonders erwähnt. Wie soll man sie einordnen?

In einzelnen Fällen kann man Grafikprogramme unter Kunstwerke und Musikprogramme unter Musikwerke einordnen. Den dokumentierten Quellcode eines Programms kann man durchaus als wissenschaftliches Sprachwerk sehen (wegen der verbalen Kommentare).

Wenn Computerwerke Sprachwerke sind, dann sind Programmiersprachen mit Sprachen gleichzusetzen. Da jedoch weder Sprachen wie Englisch und Deutsch noch irgendwelche Dialekte wie Bayerisch und Plattdeutsch Urheberrechtsschutz genießen, wären auch Computersprachen wie Basic, Pascal oder Microsoft-Basic nicht geschützt. Das gleiche gilt für Betriebssysteme wie CP/M oder MS-DOS. Das würde also bedeuten, daß jeder die Sprachen und Betriebssysteme kopieren und benützen dürfte, so wie er Esperanto oder Französisch benutzen darf. Das wiederum würde Firmen wie Digital Research und Microsoft die Existenzgrundlage entziehen, da sie doch hauptsächlich vom Verkauf selbstentwickelter Sprachen und Betriebssysteme leben.

Laut Urheberrechtsschutz sind nur »Werke« geschützt, also nur persönliche geistige Schöpfungen (§2 Abs. 2). »Persönlich« besagt, daß es von einer »natürlichen Person«, also einem Menschen, erstellt werden muß. Ein Assemblerquellcode ist also ein Werk. Was aber ist mit dem

Objektcode? Er wurde ja mechanisch mit dem Assembler erzeugt (ASM 2000 eintippen, den Rest macht der Computer und legt das Ergebnis im Speicherbereich $2000 ab). Auch ist die Übersetzung eines dokumentierten Assemblerquellcodes zum Objektcode keine schützenswerte Bearbeitung im Sinne von §3, denn Bearbeitungen müssen persönliche geistige Schöpfungen des Bearbeiters sein. So gesehen ist nur der Quellcode schutzfähig, und der befindet sich fast nie auf den raubkopierten Disks und auch meistens nicht auf der verkauften Originalsoftware-Disk.

Um nochmal auf die Bearbeitung nach §3 zurückzukommen: Wird ein Programm geknackt, dann ist dies eine persönliche geistige Schöpfung des Knackers, die auch gesondert schützenswert ist, und das laut §3 (wörtlich) »unbeschadet des Urheberrechts am bearbeiteten Werk«. Bearbeite ich also eine Originalsoftware, die eine ganze Diskseite lang ist und Kopierschutz besitzt, so, daß ich ein kurzes, einteiliges Programm habe (eventuell noch mit einem GCS (German Cracking Service) oder Section 8-Vorspann), so ist diese Bearbeitung ohne Beachtung des Urheberrechts am Ursprungswerk auch schützenswert. Auf Deutsch: Ein Knacker darf die Polizei verklagen, da sie ihm seine persönliche geistige Schöpfung beschlagnahmt hat.

Das Ganze läßt sich natürlich noch weiterspinnen: §4 UrhG besagt: »Sammlungen von Werken ... die durch Auslese oder Anordnung eine persönliche geistige Schöpfung sind (Sammelwerke), werden unbeschadet des Urheberrechts an den aufgenommenen Werken wie selbständige Werke geschützt«. Ist also eine kreativ zusammengesetzte Programmsammlung eines Raubkopierers ohne Beachtung des Rechtsschutzes der darin enthaltenen Programme eine eigene geistige Schöpfung?

Das Urheberrechtsgesetz bietet eine Fülle von Paragraphen, mit denen man derartige Gedanken noch bis ins unendliche weiterspinnen kann. Die Entdeckung eines Naturgesetzes, mathematische und logische Gesetze sowie Algorithmen werden nicht vom Urheberrecht abgedeckt, weil sie keine Werke sind. Dazu zählt zum Beispiel der Quicksort-Algorithmus (1962 von C. Hoare entwickelt.). Bestehen aber nicht alle Programme aus Algorithmen?

Wenn wir also alles oben Gesagte zusammenfassen, gilt Software (ein »Computerwerk«) also nur als urheberrechtlich geschützt, wenn es eine persönliche geistige Schöpfung darstellt und die Schutzfrist noch nicht abgelaufen ist. Wegen des Algorithmencharakters muß man jedoch vielen Programmen oder zumindest ihren algorithmischen Teilen (Rechenroutinen, Sortierroutinen etc.) die Schutzfähigkeit absprechen. Das gleiche gilt dann auch für reine Objektcodewerke, das heißt Programme, von denen nur der Objektcode aber nicht zusätzlich der Quellcode veröffentlicht wird. Wie sieht es jetzt mit kopiergeschützten Werken aus?

Ein Verlag, der ein Sprachwerk verlegt, ist aufgrund der Landespressegesetze verpflichtet, ein Vervielfältigungsstück an die entsprechende Landesbibliothek und ein weiteres Stück an die Deutsche Bibliothek in Frankfurt in lesbarer Form abzuliefern (Pflichtabgabegesetz). Dadurch soll jeder Urheber anhand des hinterlegten Sprachwerks prüfen können, ob sein eigenes Werk in urheberrechtlichem Konflikt zum fremden Werk steht. Dagegen sind Computerwerke oft in doppelter und dreifacher Ausführung unlesbar, insbesondere durch Compilierung, Codierung und Kopierschutzverfahren.

Computerwerke werden somit zu »Geheimwerken«. Sind diese jedoch überhaupt urheberrechtlich schutzfähig?

Nehmen wir als Beispiel ein Goethe-Gedicht, dessen englische Übersetzung dazu und dazu schließlich eine Anzahl von Hexzahlen, die eine Codierung des besagten Gedichts sein soll.

Würde Goethe noch leben, so wäre das Originalwerk noch voll schutzfähig. Die englische Übersetzung ist nach §3 ebenfalls ein voll urheberrechtlich schützenswertes Werk. Was ist aber jetzt mit der Übersetzung in den Code? Angenommen, ein Schwarzkopierer kopiert diese Folge von Hexzahlen, weil er denkt, diesen »ungeschützten Unsinn« kann man ohne Bedenken weiterverbreiten. Daraufhin erstattet der Urheber Anzeige. Der Urheber sagt, es handelt sich um eine Goethe-Übersetzung in einen unknackbaren Code, der Kopierer dagegen behauptet, das Werk sei eine Folge mechanischer Chiffren. Was soll nun der Richter tun, wenn er den Code nicht kennt?

Daraus ist zu schließen, daß bei Geheimwerken (ob codiert oder sonstwie geschützt) nie einwandfrei nachprüfbar ist, ob sie schutzfähig sind oder nicht. In solchen Fällen kann auch nie jemand aufgrund des Urheberrechtsgesetzes zur Rechenschaft gezogen werden (wohl aber aufgrund anderer Gesetze), denn dann könnte willkürlich jeder behaupten, seine paar Programmzeilen (vielleicht nur zwei PRINT-Befehle) wären ein schützenswertes Kunstwerk.

Ein weiterer Grund, der dafür spricht, keine Geheimwerke zu veröffentlichen, ist offensichtlich: Bei vielen Programmen und Programmpaketen kommen immer wieder die gleichen Unterroutinen und Programmteile vor, also Sortierroutinen, Druckroutinen, Routinen für Befehlserweiterungen. Wenn man diese Programme »knackt«, sind oft nicht nur die Leistungen gleich.

Man nehme also die Unterprogramme verschiedener Hersteller, kleide das Ganze in eine neue Bildschirmgestaltung und codiere und schütze das so entstandene Programm so, daß man sich Quellenangaben sparen kann.

Wer Banknoten erhält, muß selbst oder durch Experten prüfen lassen können, ob es Blüten sind. Das Pflichtabgabegesetz verpflichtet Buchverlage, die Werke lesbar zu hinterlegen, so daß auch hier festgestellt werden kann, ob es sich um Plagiate handelt. Bei kopiergeschützter Software ist es jedoch nicht möglich, so etwas festzustellen. Softwarehersteller geben ihre »Geheimwerke« mit großer Selbstverständlichkeit als Urheberwerke aus, obwohl sie es wahrscheinlich oft nicht sind. Daraus würde sich die sinnvolle Empfehlung ergeben, grundsätzlich nur diejenigen Programme unter Urheberrechtsschutz zu stellen, deren Quellcodes veröffentlicht oder öffentlich hinterlegt werden. Doch noch einmal zurück zum privaten Raubkopierer: Wie sieht es denn mit Sicherheitskopien aus?

Nach §53 und §54 UrhG sind einzelne Vervielfältigungsstücke zum persönlichen oder sonstigen eigenen Gebrauch zulässig. Die Kopien (»Werkstücke«) dürfen nicht verbreitet werden, das heißt weder getauscht, noch verkauft, noch verschenkt werden. An Freunde und Bekannte darf man sie allerdings weitergeben, also an Leute, mit denen man durch ein persönliches Band verbunden ist.

Nach einer Antwort des Bundesministers der Justiz vom 16. Juli 1968, auf eine Anfrage und deren Bestätigung durch den Bremer Schulbuchprozeß, darf man bis zu sieben Ver-



vielfältigungsstücke eines Werkes herstellen. Das heißt man darf die Kopie eines Originalprogramms (nicht einer Kopie oder Knackversion) an bis zu sieben Freunde verteilen, die diese aber nicht noch einmal kopieren dürfen. Hier taucht allerdings wieder die Frage auf, was man alles unter den Begriff »Freunde« setzt.

Nehmen wir jetzt aber einmal an, 20 Leute tun sich zusammen, um ein Originalprogramm zu kaufen. Sowohl das Urheberrechtsgesetz als auch bisherige Gerichtsurteile lassen die Frage offen, ob alle zusammen oder jeder von ihnen sieben Kopien für den eigenen Gebrauch herstellen darf. Immerhin wären das 140 legale Kopien..... also keine Raubkopien, sondern »dezentralisierte Sicherheitsbackups«.

Das gesunde Rechtsempfinden eines Richters wird sicher all die oben genannten Überlegungen über den Haufen werfen, aber trotzdem: Das deutsche Urheberrechtsgesetz weist mehr Löcher auf als der Schweizer Käse. Das UrhG schützt nur deutsche Staatsangehörige (§120). Aufgrund des Assimilationsprinzips (Welturheberrechtsabkommen, Artikel II und Berner Übereinkunft, Artikel 3) werden Ausländer im Inland wie Inländer geschützt. Aus diesem Grunde ist auch der amerikanische Copyright Act in der Bundesrepublik nicht anwendbar. Schade, denn dieses Gesetz weist weit weniger Unklarheiten auf: Computerwerke sind darin schon gesondert geregelt.

Und wenn Sie nun sagen, das wäre alles an den Haaren herbeigezogen, dann haben Sie vollkommen recht. Aber sind die Anwälte der Softwarefirmen nicht auch nur dazu da, ihr Recht an den Haaren herbeizuziehen, weil das Gesetz so viele Lücken aufweist?

(M. Kohlen/aa)

Schützer kontra Knackis

Wie kann man ein Programm sinnvoll schützen? Und wie kann man solch einen Schutz knacken? Um diese Fragen zu klären, unterhielten wir uns mit einigen Programmierern und Knackern. Sie erklärten uns ausführlich die Grundlagen des Softwareschutzes, des Knackens und des Kopierens.

Als wir uns nach den Grundlagen der Schutztechniken erkundigten, wurden wir gleich vor eine Frage gestellt: Meinten wir nun den Programm- oder den Kopierschutz? Man klärte uns auf, daß man bei Schutzmethoden folgende Unterscheidung machen muß: Als Programmschutz werden programmtechnische Maßnahmen bezeichnet, die im Computer ablaufen und das Knacken verhindern sollen. Ein Kopierschutz hingegen befindet sich auf der Diskette oder Kassette und soll das Programm vor Kopierversuchen sichern. Nur die Kombination dieser beiden Techniken ist heutzutage zum Schützen von Programmen sinnvoll.

Programmschutz

Ein Programmschutz bedeutet, daß es einem Außenstehenden fast unmöglich gemacht wird, die Arbeitsweise des Programms zu verfolgen. Hier gibt es viele Techniken, die sich allerdings auf die Maschinensprachebene beschränken. Die erste und am weitesten verbreitetste ist die Codierung von Programmteilen. Nur ein kleiner Teil des Programms liegt lauffertig vor, der Rest ist in irgendeiner Form codiert und wird erst bei Bedarf decodiert. Das allein wäre aber nicht effektiv genug: Meistens liegen die Decodierprogramme ebenfalls codiert vor, müssen also von anderen Teilen decodiert werden, und so weiter. Eine Verschachtelung von bis zu zwanzig Codier- und Decodier-Routinen, die sich vielleicht noch gegenseitig aufrufen, ist keine Seltenheit.

Fast immer wird das Codier-Prinzip mit dem Verschiebe-Prinzip gepaart: Die einzelnen Programmteile werden durch Verschieberoutinen über den gesamten Arbeitsspeicher von 64 KByte verstreut. Freaks sagen dazu auch »Spreaded Code«. Der arme Knacker hat dann praktisch keine Chance, irgendwo einen Monitor oder ein sonstiges zusätzliches Programm unterzubringen, da in jeder Ecke des Speichers ein paar Byte des Programms stehen. Weiterer Vorteil ist die damit verbundene Reset-Sperre: Man bringt Teile des Programms in Zeropage, Prozessorstack und Bildschirmspeicher unter, denn die werden bei einem Reset, so ausgetüftelt er auch sein mag, immer teilweise gelöscht.

Die nächste Schutzmethode sind die sogenannten »Illegalen Opcodes«, Maschinensprachebefehle, die offiziell gar nicht existieren. Über diese Opcodes haben wir ja schon öfter berichtet. Inzwischen sind die Softwarefirmen von der Verwendung der »Illegals« abgekommen. Aus zwei Gründen: Einerseits kennt die sowieso schon fast jeder, andererseits funktionieren die Opcodes, die man zum Schutz am besten brauchen könnte, nicht auf allen C 64. Der Grund dafür ist, das auch Fremdhersteller die 6510-CPU für Commodore in Lizenz fertigen und daß sich die einzelnen Typen nicht exakt entsprechen. Ungefähr die Hälfte der illegalen Opcodes verhalten sich auf verschiedenen Gruppen von C 64 völlig unterschiedlich. Das bringt natürlich so manchen ausgetüftelten Programmschutz zu Fall, denn was will man mit einem Programm, das nur auf einem Viertel aller C 64 läuft?

Letzte und schwierigste Methode: Selbstmodifizierender Code. Dieser Zungenbrecher bezeichnet nichts weiter, als daß sich ein Programm selbst verändert. In einem Programmteil wandelt sich dann zum Beispiel ein Sprungbefehl nach $7698 zu einem nach $4435 um oder ein LDA # wird zu einem STX. Das läßt sich sowohl sinnvoll in Programmen einsetzen, gerade in sehr schnellen Grafik-Routinen, aber auch nur zur Verwirrung des Betrachters. Es geht sogar noch schlimmer: Ein in den Interrupt eingehängtes Programm ändert perio-

disch Teile des Hauptprogramms. Dann verwandelt sich ein Sprungbefehl, der ins Leere geht, auf einmal in einen sinnvollen, und das erst direkt bevor er ausgeführt wird. Die Verwandlung selbst kann man nicht per Monitor nachverfolgen, da sie aus dem Interrupt heraus geschieht. Allerdings sei eines gesagt: Solchen Code zu schreiben ist schlimmer als ihn zu knacken. Große Softwarefirmen entwickeln deswegen diesen selbstmodifizierenden Code mit anderen Computern als dem C 64, beispielsweise mit dem IBM-PC. Ein speziell dafür ausgelegter Assembler übernimmt dann automatisch die Knochenarbeit, den Code zu erstellen.

Wenn Knacken zur Routine wird

Wenn Programme aber so gut geschützt sind, wie kann man sie dann knacken? Auch hier wurden wir sofort eines Besseren belehrt: Meistens ist ja nur der Kopierschutz so versteckt. Hat man den entfernt, braucht man sich nicht mehr um Gemeinheiten im Hauptprogramm zu kümmern. Normalerweise erstreckt sich so ein Schutzprogramm über einige wenige Teilprogramme, die von der Diskette nachgeladen werden, und ist insgesamt auch nur wenige KByte lang. Da muß man dann halt alles ausdrucken und auf dem Papier nachverfolgen, Taktzyklen zählen und sich ständig Notizen machen, welche Speicherstellen wie verändert werden. So nach und nach tastet man sich dann vor, bis man den Kopierschutz entschlüsselt vor sich liegen hat und diesen dann entfernt. Bei den neueren Schutzprogrammen muß man aber fast schon hochgradig schizophren sein, will man drei oder vier Programme, die praktisch gleichzeitig ablaufen und sich gegenseitig verändern, durchschauen.

Ein Nachteil, den manche Softwarefirmen allerdings selbst zu verschulden haben: Wird ein und derselbe Programmschutz über eine längere Zeit beibehalten, so schreiben sich manche Knacker ein Programm, das automatisch Originale dieses Herstellers entschützt. Wir bekamen als Beispiel ein Programm vorgeführt, das innerhalb von 40 Sekunden die neuen Electronic-Arts-Originale, über die nachher noch zu sprechen sein wird, voll kopierfähig macht.

Damit waren wir beim zweiten Punkt angelangt: den Kopierschutzmethoden. Der Programmschutz dient meist nur dazu, das Kopierschutzprogramm zu verstecken.

Kopierschutz

Der Kopierschutz selbst schützt nicht vor Knack- sondern vor Kopierversuchen (oder soll dies zumindest). Wir erfuhren die gängigsten Verfahren des Kassetten- und Diskettenschutzes.

Bei Kassetten ist es eigentlich ganz einfach: Man entwickelt ein eigenes Aufzeichnungsformat und die dazu passenden Leseroutinen. Neben dem Kopierschutz ist dann auch gleichzeitig ein Turbo-Lader realisierbar. Ein brandneues Schutzsystem ist so ausgetüftelt, daß das Laden des Programmes unmöglich ist, wenn der C 64 minimal verändert worden oder ein zusätzliches Peripheriegerät (Floppy, Drucker oder auch nur Modul im Expansionport) angeschlossen ist. Dann schwankt die Spannungsversorgung des C 64 und somit auch die Motorsteuerung des Recorders minimal, und das haarscharfe Timing der Leserroutine bricht zusammen.

Solche Kassetten mit zwei Hi-Fi-Tapedecks zu überspielen klappt meist nicht, denn die sind zu gut und zerstören das Signal beim Überspielen durch Frequenz- und Dynamikkorrekturen. Mit billigen Recordern geht's auch nicht, weil sich da die Motorschwankungen beim Überspielen addieren. Und ein Kassettenkopierprogramm für den C 64, das alle möglichen Formate kopiert, kann aus verschiedenen technischen Gründen nicht geschrieben werden. So ist beispielsweise die Motorsteuerung durch den Computer nicht präzise genug. Kassettenprogramme sind gegen Kopierversuche im großen und ganzen besser schützbar als Diskettenprogramme. Deswegen wird englische Software hauptsächlich auf Kassette angeboten. In Amerika und Deutschland ist hingegen die Floppy so weit verbreitet, daß sich Programme nur auf Diskette in genügend großen Mengen verkaufen lassen.

Um einiges vielfältiger sind daher die Methoden des Diskettenschutzes. Die gängigsten und aktuellsten Methoden wurden uns im Schnelldurchgang vorgestellt:

Da wären erst einmal die »Errors«, künstlich auf die Diskette aufgebrachte Lesefehler. Meist werden dann auch noch Daten in den fehlerhaften Blöcken versteckt. Diese Methode ist aber viel zu bekannt, und die meisten Fehler werden von jedem zweitklassigen Kopierprogramm einfach mit übertragen. Ein Lesefehler besonderer Art sind die Killertracks, Tracks, die komplett mit Synchronmarkierungen vollgeschrieben wurden. Will man normal auf einen solchen Track zugreifen, hängt sich die Floppy unweigerlich auf. Eine Zeitlang waren auch physikalische Fehler im Gespräch: Die Diskette wird an einigen Stellen beschädigt und dann wird versucht, diese Stelle zu beschreiben. Doch dieses Verfahren ist sehr kompliziert und teuer in der Herstellung, sollen alle Disketten gleich defekt sein. Es wird deswegen nur von kleinen Firmen für Programme verwendet, die in nicht allzuhohen Stückzahlen auf den Markt kommen. Sowas ist natürlich nicht softwaremäßig kopierbar, aber so mancher hat es schon durch optische Kontrolle des Originals und gezieltes Zerstören der Kopie geschafft.

Interessanter als Errors sind da schon die Blockheader-Manipulationen. In einem Blockheader, der jedem Datenblock vorangeht, sind für den Disk-Controller wichtige Daten abgelegt. Deren Manipulation kann zu vielfältigen Ergebnissen führen: Vertauschung der Reihenfolge der Sektoren auf einem Track, Blöcke die doppelt mit unterschiedlichen Daten vorhanden sind, illegale Track- und Sektor-Nummern auf legalen Positionen, Vertauschung zweier Tracks miteinander und so weiter...

Die meisten dieser Verfahren führen dazu, daß Teile der Diskette nur noch mit speziellen Programmen gelesen werden können. Die sind dann natürlich, wie oben beschrieben, versteckt.

Nächster Ansatzpunkt sind Halbspuren (Halftracks) und illegale Tracks. Der Schreib-/Lesekopf der 1541 kann in Halbspurschritten bewegt werden. Allerdings ist es nicht möglich, eine Halbspur zu beschreiben, ohne die beiden angrenzenden Tracks teilweise zu löschen. Aber immerhin kann man so, verzichtet man auf die Tracks 12 und 13, den Track 12,5 beschreiben und ihn entweder als Track 12 oder Track 13 verwenden. Einer von beiden geht dabei völlig verloren. Wird beim Lesen von Sektoren dieses Tracks abgefragt, um wieviele Halbspurschritte sich der Kopf bewegen mußte um diesen Track zu erreichen, lassen sich Original und Kopie voneinander unterscheiden.



Illegale Tracks sind die Tracks 36 bis 42. Diese Tracks sind, obwohl offiziell nicht vorhanden, beschreib- und lesbar, allerdings auch nur mit speziellen Programmen und bei Disketten guter Qualität.

Die letzte einfache Schutzmöglichkeit ist das Manipulieren von Lücken. Solche Lücken befinden sich immer zwischen Blockheader und Datenblock und umgekehrt. Man kann die Länge dieser Lücken verändern oder aber Daten in ihnen verstecken.

Der ewige Wettlauf

Da sich alle diese Verfahren aber, wie später noch beschrieben wird, recht einfach kopieren lassen, fahren die Softwarefirmen in letzter Zeit ein sehr schweres Geschütz auf: Fremdformate. Die Datenaufzeichnung auf der 1541 unterliegt vielen Spielregeln. Wenn man diese nicht nur teilweise übertritt, sondern völlig außer Kraft setzt, so erhält man Ansammlungen von Bytes auf der Diskette, die das DOS der 1541 nicht mehr verarbeiten kann. Diese Daten können dann nur noch mit Spezialprogrammen gelesen werden. Eine Möglichkeit eines Fremdformates wäre zum Beispiel die Auflösung der Sektor-Struktur auf einem Track. Hinter einer einzigen Synchronmarkierung befinden sich dann direkt aufeinanderfolgend die Datenbytes. Wenn man hier geschickt arbeitet, kann man alle Kopierversuche mit einfachen Kopierprogrammen unterbinden. Doch haben Programmierer inzwischen auch hier Mittel und Wege zum Kopieren gefunden.

Einen einzigen Kopierschutz gibt es bisher, der garantiert nicht rein softwaremäßig und ohne Umbau der 1541 kopiert werden kann. Dieser Schutz wird bisher nur von Electronic-Arts verwendet. Diese Firma kopiert ihre Disketten mit einer speziellen Kopiermaschine, die einen zu breiten Tonkopf hat, so daß sie immer zwei Tracks gleichzeitig beschreiben kann. Daraus folgt, daß die Tracks 34, 34,5 und 35 völlig identisch und parallel zueinander sind. Beim Lesen von Blöcken des Tracks 34 kann also der Kopf in Halbspurschritten nach innen und wieder zurück nach außen bewegt werden, ohne daß Leseprobleme auftreten. Dieser an sich einfache Schutz ist nicht mit der normalen 1541 kopierbar, da diese ja im eigentlichen Sinne nicht halbspurfähig ist.

Um bei den Firmen zu bleiben: Wir wollten wissen, wie die anderen großen Firmen schützen. Schlechtester Schützer ist augenblicklich Activision, die immer noch mit Errors arbeitet. Broderbund benutzt neuerdings ein einfaches eigenes Format auf den Tracks 36 und 37 gekoppelt mit Killertracks. Epyx arbeitete bisher mit Lücken, stellt aber gerade auf ein neues Verfahren um. Data Becker verwendet gefüllte Lücken und minipulierte Header auf den Tracks 1 bis 40, eine gefährliche Sache, weil schon damit leicht verstellte 1541-Laufwerke nicht mehr zurecht kommen. Datasoft schließlich treibt den größten Aufwand mit einem eigenen ausgeklügelten Format auf fast allen Tracks.

Kopierprogramme, die jeder haben will

Jetzt fragten wir natürlich noch nach den gängigen Kopierverfahren und -programmen.

Beschränkt man sich auf Kopierprogramme, die einen Kopierschutz mitkopieren sollen, gibt es derzeit drei Prinzipien. Dabei kann jedes Verfahren ganz einfach auf Halbspuren und illegale Tracks ausgeweitet werden, so daß wir diese nicht weiter beachten.

Beim ersten Verfahren, dem »Header-Copy«, werden Blockheader und Datenblock, vielleicht noch die Lücken kopiert. Das Verfahren kopiert viele aber nicht alle Fehler und die meisten Header-Manipulationen, aber nicht Killertracks oder gar Fremdformate. Ein Vertreter dieser Gattung ist »Turbo-Nibbler«, ein weiterer »Superclone«. Allerdings ist diese Methode inzwischen veraltet, da die kleinen Details, die diese Kopierprogramme zwangsläufig übersehen müssen, von den Softwarefirmen heutzutage gezielt eingesetzt werden.

Die zweite Gruppe von Kopierprogrammen sind die Synchronmarken-orientierten Programme (Sync-Copy). Hier wird nicht mehr zwischen Blockheader, Datenblock und Lücke unterschieden. Der Blockheader und der Datenblock werden immer von einer Synchronmarkierung, kurz Sync genannt, eingeleitet. Sync-orientierte Programme kopieren nun immer von Sync zu Sync, ohne sich darum zu kümmern, was sie eigentlich kopieren. Mit diesem Verfahren werden alle erzeugbaren Fehler, alle Blockheader- und Lückenmanipulationen und die einfachen Fremdformate kopiert. Dieses Verfahren versagt nur, wenn gar keine Syncs vorhanden sind oder wenn sich mehr als 1000 Byte zwischen zwei Syncs befinden. Dann reicht die Kapazität des Floppy-Pufferspeichers nicht mehr aus, die Daten zwischen den Syncs zu speichern. Bekanntester Vertreter dieser Gruppe ist »Doubble Image«. Einziger Nachteil dieses Verfahrens ist, daß die Disketten mindestens siebenmal gewechselt werden müssen, weil hier viel mehr Daten gelesen und geschrieben werden als normalerweise üblich.

Der Sieger steht noch nicht fest

Das letzte Verfahren ist noch nicht ausgereift, es handelt sich um »Burst-Copys«. Beim Burst-Verfahren wird auf keine Markierung auf der Diskette mehr Rücksicht genommen, ein Track wird während einer einzigen Umdrehung komplett gelesen und geschrieben, egal wie sein Inhalt aussieht. Damit würde praktisch alles kopiert werden, bis auf den Electronic-Arts-Schutz. Im Augenblick gibt es noch kein voll funktionsfähiges Burst-Copy, da die 1541 eigentlich nicht dafür ausgelegt ist. Die eine Alternative wäre, mit einem System zu arbeiten, das den seriellen Bus extrem beschleunigt. Die bisherigen Vertreter, Turbo Access und SpeedDos, sind hier hart an der Grenze der benötigten Geschwindigkeit. Die andere Möglichkeit ist eine RAM-Erweiterung der 1541 um mindestens 10 KByte, in der ein Track komplett zwischengespeichert werden kann, bevor er über den seriellen Bus geht.

Um auch die letzte Bastion, den Electronic-Arts-Schutz, zu kopieren, müßte man dann noch die Mechanik der 1541 gegen die eines 80-Spur-Laufwerkes tauschen oder aber das Indexloch, das auf der 1541 nicht verwendet wird, mit einer Lichtschranke nachrüsten, um parallele Halbspurformatierungen zu ermöglichen. Mit einem solchen hochgepäppeltem Laufwerk wäre wohl kein Kopierschutz mehr sicher.

Der Wettlauf zwischen Schutz- und Knackprogrammen geht also weiter. Die Frage, ob es wirklich keinen perfekten Programm- oder Kopierschutz gibt, kann man nur mit einem Augenzwinkern beantworten: »Ein schlechtes Programm, das keiner knacken oder kopieren will!«

(B. Schneider/aa)

Die Ex-Knacker — wo sind sie geblieben?

Die Szene hat sich verändert. Der Raubkopierermarkt wird heute von anderen Namen beherrscht.

Damals noch gab es innerhalb der geschlossenen Gesellschaft der Raubkopierer eine höhergestellte Klasse der »Superheroes of Cracking« (zumindest erweckten sie den Eindruck einer exklusiven Gesellschaftsschicht, an die ein normaler Computerfreak nicht so leicht herankam). Dazu gehörten so schillernde Gestalten wie Antiram, 1103, Oleander, Mister O, Jedi, Kotzbrocken und andere. Doch von diesen Pseudonymen sieht man heute nichts mehr. Sind die alten Knacker den Säuberungsaktionen der Softwarefirmen zum Opfer gefallen, haben sie grundlos aufgehört zu knacken oder arbeiten sie mittlerweile ganz legal für die Softwarefirmen?

Nach einigen Recherchen ist es uns gelungen, diese Leute zu befragen (Ex-Knacker reden ungern über ihre Vergangenheit). Entscheidendes Ergebnis: Der Polizei ist keiner zum Opfer gefallen. »Dazu haben wir uns rechtzeitig aus dem Raubkopierermilieu entfernt«, sagt einer von ihnen. »Die meisten von uns sind schon ausgestiegen, bevor die Firmen mit der intensiven Verfolgung der Softwarepiraten begannen.« Also war das nicht der ausschlaggebende Grund, damit aufzuhören? — »Nein. Sicher ist es gefährlicher als früher, aber wenn wir wollten, könnten wir jetzt noch immer so weitermachen wie damals; Wir haben keine Softwarelisten verschickt, und andere Beweise, wer wir sind — oder besser waren — gibt es nicht. Und auf ein paar zweifelhafte Zeugenaussagen kann sich kein Gericht stützen, eher hätten diese "Zeugen" sehr bald eine Verleumdungsklage am Hals.« Warum habt Ihr dann aufgehört zu knacken? — »Da gibt es viele Gründe, die alle zusammenspielen. Außerdem knacken wir manchmal noch immer, aber nicht mehr für andere Leute. Wir sind lediglich aus der Raubkopiererszene ausgestiegen.« Also gut, warum seid Ihr ausgestiegen? — »Für die meisten von uns war es einfach langweilig geworden. Irgendwann wiederholen sich die Schutzmechanismen, die Softwarefirmen lassen sich nichts Neues mehr einfallen. Da wären unsere Kopierschutzmethoden sicher besser. Man muß aber zugeben, daß das Knacken einen gewissen Lerneffekt mit sich brachte. Man lernt dabei eine Menge über Programmiertechniken. Mittlerweile sind wir in der Lage, auch selbst professionelle Software zu schreiben.«

Knacken als Lehre und Anschauungsunterricht für professionelle Software-Entwickler — ein neuer Aspekt. Und tatsächlich entwickeln einige von den einstigen Helden des Knackertums professionelle Software und arbeiten für große und bekannte Firmen: »Eigene Software schreiben macht mittlerweile mehr Spaß, als in fremder Software herumzuwühlen. Auch das ist ein Grund gewesen, aufzuhören. Und schließlich sind einige von uns auch mit anderen Dingen (Studium, Arbeit, Bundeswehr) beschäftigt und haben keine Zeit mehr für ihren Computer. Andere wiederum streben in die höheren Ebenen und planen, selbst einen Computer zu bauen.« Wer von den Ex-Knackern in das legale Computer-Business einsteigen wollte, hat es geschafft. Es existieren keine Jedis, Antirams, KBRs oder 1103s mehr. Die Szene sieht anders aus als damals. Wir wollten von den jetzt »legal« gewordenen erfahren, was sie denn von der gegenwärtigen Szene im Vergleich zur damaligen halten. — »Damals, als wir aktiv waren, hatten die Kopien einfach eine bessere Qualität. Zur Zeit sind nur noch wenige Leute aktiv in diesem Gebiet tätig. Und die verstehen es nicht, sauber zu knacken. Da wird zum Beispiel ein Spiel, das nur 4 KByte lang ist, mit 202 Blocks auf Disk gespeichert, also einfach Reset und den ganzen Speicher SAVEn. So dilettantisch haben wir nicht gearbeitet. Allerdings hatten wir nicht geknackt, um möglichst schnell und viel zu verbreiten, sondern aus Spaß am Knacken und dem Erfolgserlebnis, **gute** Arbeit geleistet zu haben. Außerdem hatten wir die Kopien nicht gleich an jeden weitergegeben, sondern erst einmal eine Weile vor der Allgemeinheit zurückgehalten. Das gab uns zumindest das Gefühl, den legalen Händlern eine Chance zu lassen. Dieses Gefühl von Loyalität zu den Händlern hat heute keiner mehr; wahrscheinlich, weil diese Leute nie mit Händlern zusammenarbeiteten (wie wir manchmal) und weil sie nie selbst Software schreiben.« Wieso diese Loyalität zu den Händlern? — »Wir sind zum Teil selber welche...«

(M. Kohlen/aa)

Begriffsbestimmungen

Da sich nun mittlerweile jeder, der einen Akustikkoppler besitzt, Hacker nennt (was nicht richtig ist), und von verschiedenen Zeitschriften jeder Raubkopierer als »Cracker« bezeichnet wird, wollen wir einmal Ordnung ins System bringen:

Raubkopierer
Alle, die unerlaubt Software kopieren. Anderes Wort dafür: Schwarzkopierer.

Hacker
Dieser Begriff ist im internationalen Sprachgebrauch mehrdeutig. In Deutschland bezeichnen wir als Hacker diejenigen, die sich mit DFÜ beschäftigen, wirklich Ahnung von der Sache haben und schon mal ein »Password« knacken können.

International werden als Hacker öfters auch diejenigen bezeichnet, die den ganzen Tag (oder die ganze Nacht) auf ihrem Computer herumhacken und nichts anderes mehr im Sinn haben.

Cracker
Cracker oder Knacker sind Leute, die den Kopierschutz aus Programmen entfernen, um kürzere, bessere und leichter vervielfältigbare Software daraus zu machen. Nicht jeder Knacker ist ein Raubkopierer, sofern er die Sache im stillen Kämmerlein betreibt und keine Raubkopien weiterverbreitet.

Mugger
Dieser Begriff wird von der amerikanischen Fachpresse für diejenigen verwendet, die zwar nicht knacken können und auch keine Ahnung vom Programmieren haben, die aber den Firmen den meisten Schaden zufügen, weil sie mit Hilfe von Kopierprogrammen alles weiterverbreiten, was sie in die Finger kriegen.

Dealer
Die übelste Sorte von Schwarzkopierern: Leute, die professionell am Verkauf von Raubkopien verdienen.

Bei der Recherchierarbeit für diesen Artikel fiel uns beim Sichten von »geknackten« Programmen immer wieder ein Name auf: Section 8. Da wir hier eine größere Gruppe von Knackern in fast schon professionellem Stil vermuteten, versuchten wir, Kontakt mit Section 8 aufzunehmen. Schließlich konnten wir sie zu einem längeren Telefoninterview überreden, deren wichtigsten Aussagen wir hier in geraffter Form wiedergeben wollen:

Zuerst einmal wollten wir natürlich wissen, wer Section 8 eigentlich sei. Es ist eine Gruppe von vier Leuten zwischen 15 und 24 Jahren. Die meisten gehen ganz normal zur Schule. Bis vor kurzem beschäftigten sie sich nur mit dem Commodore 64, inzwischen hat sich einer noch einen Apple zugelegt. Die Frage nach dem typischen Tagesablauf beantworteten sie wie folgt: »Der Vormittag steht im Zeichen der Schule. Der erste Weg am Nachmittag führt zum Briefkasten, danach wird kopiert und geknackt«. Andere Hobbys, die nichts mit Computer zu tun haben, haben die vier überhaupt nicht.

Einige »statistische« Angaben machten sie uns auch gleich. Pro Woche werden mindestens fünf, in Spitzenzeiten über zwanzig Programme geknackt. Die durchschnittliche Zeit, um ein Programm komplett zu bearbeiten, ist etwa 45 Minuten. Dabei wird das Original-Programm analysiert, der Kopierschutz entfernt, das Programm umkopiert. Außerdem wird noch der Name der Gruppe im Titelbild des Programmes oder im Programmcode versteckt. Diese für uns erstaunlich geringe Durchschnittszeit resultiert daraus, daß Firmen ihren Schutz selten ändern, so daß man, ist erst einmal ein Programm einer Firma geknackt, für die weiteren nur noch wenige Minuten benötigt. »Harte Brocken«, Programme also, bei denen man ganz von vorn anfangen muß, benötigen dank der bisher gesammelten Erfahrungen sehr selten mehr als drei bis vier Stunden. Das Knacken ist fast schon eine Routinearbeit geworden. Dies brachte auf unserer Seite die Frage nach dem Warum auf.

Warum wird geknackt?

Dies wurde uns mit mehreren Argumenten beantwortet. »Erstens einmal macht es ganz einfach Spaß, die Softwarefirmen zu »bescheißen« (Originalzitat Section 8). Der hauptsächliche Grund liegt aber in den hohen Softwarepreisen verborgen.



An professionelle Raubkopierer ist schwer ranzukommen. Dennoch gelang uns zumindest ein Telefoninterview mit Section 8, einer der bekanntesten Raubsoftvereinigung Deutschlands.

Meistens sieht es so aus, daß eine etwas größere Gruppe von etwa zehn Leuten jede Woche Originale einkauft.

Diese werden dann geknackt, jeder erhält eine Kopie von jedem, behält aber ein Original. So zahlt jeder effektiv nur ein Zehntel von dem, was die Software im Laden kosten würde. Außerdem sind bei dieser Verfahrensweise stets die Anleitungen vorhanden oder können fotokopiert werden, so daß auch professionelle Software, wie Textverarbeitungen und Datenbanken, beliebte Knackobjekte sind. Kurze Zeit später gehen die Programme an weitere Leute. Auch hier wurden uns ein paar Zahlen genannt: Einer der vier »beliefert« jede Woche etwa 25 Leute im Inland und 50 im Ausland. Die hohen Portokosten und das Geld für die Disketten werden durch »freiwillige Geldspenden« wieder hereingeholt. Direkt verkauft wird aus Sicherheitsgründen nicht. Daß dann wenige Wochen später sehr viele Leute die Software haben, stört Section 8 nicht. Geknackt wird hauptsächlich damit man die Software selber hat. Daß sie dann »rumgeht«, ist mehr oder minder ein nützlicher Nebeneffekt und gar nicht mal so sehr das eigentliche Ziel. Natürlich ist es aber ein gutes Gefühl, wenn man weiß, daß Unmengen von Commodore 64-Besitzern Software bei sich rumstehen haben, die von Section 8 kommt. Der »Ruhm« ist erwünscht. Section 8 möchten die bekanntesten Knacker in Deutschland werden, aber das ist eben, wie schon gesagt, eine Nebenwirkung.

Unsere Zwischenfrage, warum denn offensichtlich nicht das Argument der Softwarefirmen ziehe, daß Raubkopien und Knackversionen den Firmen ungeheuere Verluste zuweisen, und daß so gezwungenermaßen die Qualität der Software sinken müßte, wurde klar beantwortet: »Von dem was jede Woche kommt, ist

mindestens ein Viertel in sehr guter Qualität«. Seit Section 8 knackt, konnten sie keinen Qualitätsverlust, sondern nur einen Anstieg feststellen. »Die großen Firmen könnten sicherlich nicht ein Jahr überleben, würden sie wirklich so schlimm dran sein, wie sie immer behaupten. Die hohen Softwarepreise sind also garantiert immer so kalkuliert, daß die Verluste vollkommen ausgeglichen werden. Bei manchen Firmen sei sogar das Gegenteil der Fall, sie werden durch die Knacker regelrecht unterstützt. Das markanteste Beispiel ist Commodore selbst. Diese Firma solle ja nicht glauben, daß sich der C 64 so gut verkauft, weil er der beste Computer derzeit ist, sondern weil Jugendliche hier am leichtesten umsonst an wirklich gute Software, hauptsächlich wohl Spitzenspiele, herankommen können. Und die, die am lautesten klagen, Data Becker, bringen gleichzeitig ihre Trainingsbuchreihe heraus, die sich durchgehend besser verkauft, als die behandelte Software selbst. Und Data Becker wird wohl auch ganz genau wissen, daß die meisten der Käufer die Bücher nur deswegen kaufen, weil sie kein Original und somit keine Anleitung haben, sondern eben beispielsweise eine Section 8-Version. Wenn die Knacker nicht wären, gäb's diese Bücher und die damit erzielten Gewinne auch nicht.«

So beschaffen sie Originale

Einige weitere Argumente, die Section 8 für das Knacken anführte: »Knacker sind durchgehend schneller als die Softwarefirmen«. Uns wurde als Beispiel gesagt, daß man schon sehr bald mit der Knack-Version von »Karateka« von Broderbund für den C 64 rechnen könne, zumindest Wochen bevor Ariola auch nur ein einziges Original liefern kann. (Da das Interview fast zwei Monate vor Erscheinen dieser Ausgabe geführt wurde, dürfte Karateka schon bei einigen als Section 8-Version vorhanden sein). Wir wollten hier natürlich sofort wissen, wie das angehen kann. So erklärte uns Section 8, wie sie normalerweise an Software herankommen. Neben dem harten Kern der angesprochenen vier Leute hat Section 8 viele Bekannte im Ausland, insbesondere in England und Amerika, wo die Software schon Wochen vorher als auf dem deutschen Markt erscheint. Von denen läßt man sich die Kataloge der Firmen zuschicken, bestellt dann dort die Software und erhält sie per Luftpost. Auch Bekannte, die mal geschäftlich ins Ausland reisen, werden mit dem Einkauf von Software beauftragt, dafür erhalten sie dann immer die neuesten Knack-Versionen. Teilweise hat man sogar Kontakt zu Mitarbeitern von Firmen. Da ergab sich natürlich die Frage, ob man hier nicht schon von einem internationelen Software-Ring sprechen könnte. Von einer solchen Bezeichnung distanzierte man sich aber. Es ist nicht so, daß es hier eine straffe, hierarchische Organisation gebe, die damit Geld verdienen würde. Es werden einzig und allein die Unkosten wieder hereingeholt, und sonst läuft das Ganze halt wie unter guten Bekannten ab.

Warum nicht selber schreiben?

Nachdem die Frage des Warum damit sehr ausführlich geklärt war, kamen wir noch einmal auf das Knacken an sich zurück. Wir wollten wissen, warum denn solche Experten des Commodore 64 nicht lieber selber Software schreiben. Gefallen würde ihnen das schon, weil sich davon (trotz Knackern) sehr gut leben ließe. Aber das wäre den Section 8's einfach zu viel Aufwand. In ein gutes, umfangreiches Programm müssen mehrere Monate Arbeit hineingesteckt werden. Das wäre ihnen einfach zu langweilig. Es fehlen hier die schnellen Erfolgserlebnisse.

Trotzdem werden Programme geschrieben, meist Knack-Utilities oder Kopierprogramme. Gerade arbeite man an einem neuen Vorspann, der vor jede Section 8-Version gesetzt werden soll (siehe Bild).

Außerdem werden Programme verbessert. So gibt es neuerdings von Oceans »Daley Thompsons Decathlon« eine Section 8-Version, die zu zweit gegeneinander gespielt werden kann (beim Original ist dies nicht der Fall). Bei dieser Gelegenheit wurden Sound und Grafik auch etwas ausgebessert. Solche Projekte werden aber nur in Angriff genommen, wenn es gerade nichts zu knacken gibt.

Weitere Argumente, die für Section 8 gegen das Schreiben von Software sprechen: Es gibt viel zu viele gute Programme für den C 64, da muß man, um einen Renner zu landen, noch mehr Arbeit investieren. Da warten sie lieber auf einen Computer, der noch mehr Möglichkeiten bietet, wie beispielsweise der Atari 520 ST oder der Amiga. Wenn man da sofort einsteigt und gute Software liefern kann, lohnt sich das viel eher, kann man sich schneller einen »Programmierer-Namen« machen.

Da wir gerade beim Thema neue Computer waren, fragten wir nach der Meinung zum 128er. Für Section 8 ist das Gerät eine Totgeburt, denn es bietet gegenüber dem 64er auf den Gebieten Grafik und Sound rein gar nichts Neues. Das bessere Basic ziehe unter Maschinenspracheprogrammieren so und so nicht. CP/M wird ihrer Ansicht nach spätestens mit GEM sterben. Sie werden sich also wahrscheinlich keinen zulegen. Wie gesagt, wartet man lieber auf den Atari 520 oder den Amiga, die wirklich Neues bieten.

Die Frage nach den Konsequenzen

Zurück zum Thema Knacken: Wir wollten wissen, ob man sich über die rechtlichen Konsequenzen im klaren sei, falls man erwischt werden sollte. Section 8 hält es aber für extrem unwahrscheinlich, daß jemals die Polizei vor ihrer Haustür steht. Und sollte das je der Fall sein, so wird man zwar stapelweise Originale finden und auch einige aktuelle Raubkopien, maximal vier bis fünf Disketten voll, aber keinerlei Hinweise, daß dort kopiert und geknackt wird. Bei ihnen sieht es ab und zu schon so aus, wie man es sich vielleicht vorstellt: Haufenweise Disketten, Computer, Notizpapier und Bücher, und vor allen Dingen Unordnung. Aber das ist eher selten, meist sieht es so aus, wie bei jedem anderen Commodore 64-Besitzer. Man ist sich sehr sicher, daß niemand nachweisen kann, wer Section 8 ist. Und selbst wenn es zu irgendwelchen Maßnahmen kommen sollte, so besorgt man sich halt einen guten Rechtsanwalt, der sich mit der Materie auskennt. Die Rechtslage ist laut Section 8 nämlich derart verworren, daß man sich auch hier ziemlich sicher fühlen darf.

Um langsam zu einem Ende zu kommen, wollten wir noch wissen, welche Firmen nach Ansicht von Section 8 am besten schützen. Bei Disketten seien dies Datasoft und vielleicht noch Electronic Arts, bei den Kassetten hauptsächlich Anirog und andere englische Firmen.

Als allerletztes wollten wir noch etwas über die Zukunftspläne von Section 8 hören. Wann sie mit Knacken aufhören werden, weil es ihnen zu langweilig wird, ist augenblicklich noch nicht abzusehen. Man wird zumindest in nächster Zeit Section 8 noch etwas bekannter machen wollen: So sind gerade eigene »Section 8 International«-Aufkleber und -Buttons in Produktion. Und direkt nach dem Auflegen des Telefonhörers würden sie sich einem Eilpäckchen aus Österreich widmen, das die Post gerade vorbeigebracht hat.

(M. Kohlen/B. Schneider/aa)

Grafikeingabegeräte: Was ist das? Wie funktionieren sie?

Viele, die einen Trackball, ein Grafiktablett oder einen Lichtgriffel besitzen, werden sich sicherlich schon einmal die Frage gestellt haben, wie solche Geräte, rein von der technischen Seite her, funktionieren.



Bild 1. Grafische Darstellung des Innenlebens eines Trackballs.

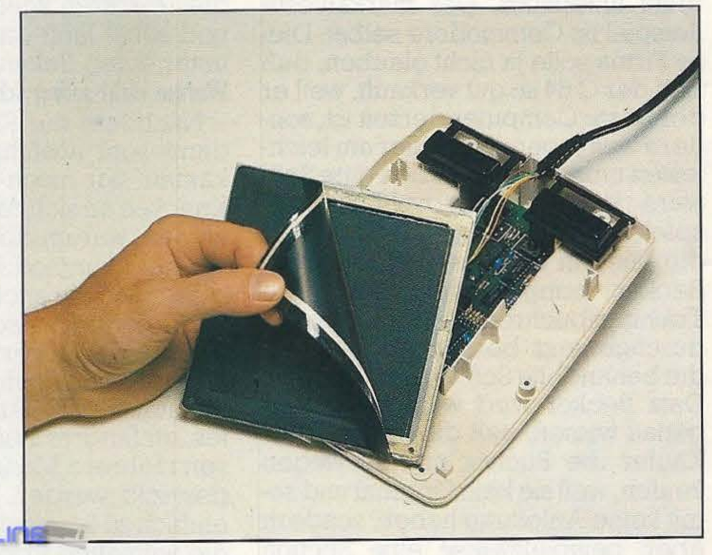


Bild 3. So sieht ein Grafiktablett von innen aus.

Es ist doch immer wieder erstaunlich, wie zum Beispiel ein Lichtgriffel in der Lage ist, Spuren auf dem Bildschirm zu hinterlassen, obwohl sich auf dem Schirm keinerlei Sensoren befinden. Auf diese und viele anderen Fragen soll hier eine Antwort gefunden werden. Der versierte Elektronikbastler ist nach dem Lesen dieses Artikels in der Lage, sich die hier beschriebenen Eingabegeräte selbst zu entwickeln. Es wird jedoch keine Bauanleitung geliefert, sondern nur die dafür erforderlichen Grundlagen.

Der Trackball — Ein Ersatz für den Joystick

Wir wollen mit dem Leichtesten anfangen, dem Trackball. Bei ihm handelt es sich um ein Eingabegerät, das die gleichen Aufgaben erfüllt wie ein Joystick. Im Computer werden die gleichen Register angesprochen. Der Unterschied liegt darin, daß man keinen »Knüppel« in der Hand hat, um Objekte über den Bildschirm zu bewegen, sondern an einer »Kugel« dreht. Das Erstaunliche und Unverständliche an diesem Eingabegerät dürfte wohl die Tatsache sein, daß es in keiner Richtung einen »Anschlag« gibt, wie zum Beispiel ein Lautstärkeregler am Radiogerät. Man kann die Kugel drehen und drehen und .... und kommt zu keinem Ende. Das Objekt bewegt sich wie erwartet kontinuierlich über den Bildschirm. Um die Funktionsweise eines solchen Trackballs zu verstehen, muß man ihn zerlegen (Bild 1).

Es ist zu erkennen, daß die Drehbewegung der Kugel auf drei Achsen übertragen wird. Eine dieser Achsen hat nur die Aufgabe eines Stabilisators. An den Enden der anderen beiden Achsen befinden sich zwei Schlitzscheiben. Da die beiden Achsen um 90 Grad versetzt sind, können Bewegungen in alle Richtungen erkannt werden, wobei jede Achse die Bewegungskomponenten in je zwei Hauptrichtungen feststellt. Die Schlitzscheiben regen beim Drehen je zwei Lichtschranken zu Impulsen an, die mit einem Schmitt-Trigger stabilisiert werden. Es stellt sich natürlich sofort die Frage, wodurch erkennt der Computer, ob die Kugel in positiver oder negativer Richtung gedreht wird? Denn die erzeugten Impulse sind in beiden Fällen die gleichen. Genau aus diesem Grund befinden sich jeweils zwei Lichtschrankenpaare im Trackball. Jedes Paar erzeugt für x- oder y- Richtung beim Drehen zwei Rechteckschwingungen, die um 90 Grad versetzt sind (Bild 2). Bezieht man sich auf Schwingung 1, so überprüft die im Trackball eingebaute Elektronik bei jeder positiven Flanke (Anstieg der Spannung von Null auf Eins) der Schwingung 2. Wird die Trackball-Kugel in positive x-Richtung gedreht, so beträgt der Spannungspegel der Schwingung 2 bei jeder positiven Flanke der Schwingung 1 Null. Wird die Trackball-Kugel dagegen in negative x-Richtung gedreht, so beträgt der Spannungspegel der Schwingung 2 bei jeder positiven Flanke der Schwingung 1 Eins. Das liegt ganz einfach daran, daß die positive Flanke in positiver x-Richtung einer negativen Flanke in negativer x-Richtung entspricht und anders herum. Sie können es gleich selbst einmal ausprobieren. Nehmen Sie dazu einen Bleistift und schieben ihn parallel zur gestrichelten Linie horizontal über das Papier in die positive x-Richtung (Bild 2). Achten Sie dabei

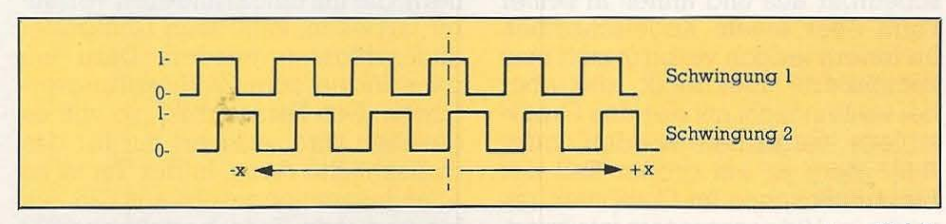
nur auf Schwingung 1. Sobald der Bleistift eine positive Flanke berührt, sehen Sie sich die Schwingung 2 an. Der Spannungspegel beträgt wie erwartet Null. Nun schieben Sie den Bleistift in die negative Richtung und Sie werden feststellen, daß der Spannungspegel der Schwingung 2 Eins beträgt, wenn Sie auf eine positive Flanke der Schwingung 1 stoßen. Dieser Unterschied läßt sich elektronisch sehr leicht auswerten, indem jedesmal wenn eine positive Flanke der Schwingung 1 auftritt, zwei Flip-Flops in Abhängigkeit von Schwingung 2 gesetzt und bei einer negativen Flanke der Schwingung 1 gelöscht werden.

Die Ausgänge dieser beiden Flip-Flops werden unmittelbar auf die Porteingänge des Joysticks gelegt. Zu beachten ist allerdings, daß die Ausgänge der Flip-Flops in Ruhelage, das heißt wenn der Trackball nicht gedreht wird, positives Spannungspotential haben.

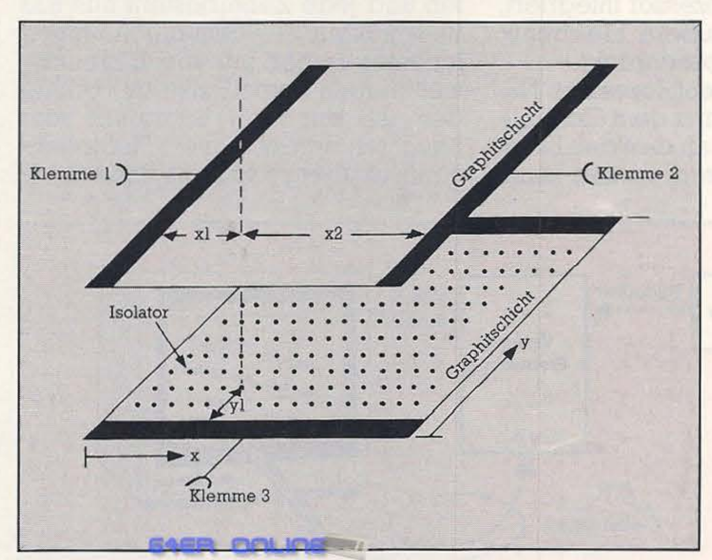
Das Grafiktablett – Ein analoges Eingabegerät

Außer dem Trackball, der wie schon erwähnt in seiner Funktion einem Joystick ähnelt und auch genauso angesprochen wird, gibt es noch das Grafiktablett. Hierbei handelt es sich, im Gegensatz zum Joystick oder Trackball, um ein analoges, also kontinuierliches Eingabegerät. Es existieren nicht mehr nur die beiden Zustände Null und Eins beziehungsweise 0 Volt und 5 Volt, sondern die Ausgangsspannung des Grafiktabletts kann kontinuierliche Werte zwischen 0 Volt und 5 Volt annehmen. Ein Computer kann aber nur zwischen 5 Volt (gleich Eins) und 0 Volt (gleich Null) unterscheiden. Deshalb muß diese analoge Spannung in ein digitales Signal gewandelt werden. Eine solche Wandlereinheit, auch A-D-Wandler genannt, ist im C 64, genauer gesagt im Soundcontroller, gleich zweimal enthalten. Die Anschlüsse am Joystickport heißen Pot-x und Pot-y. Genau an diese beiden Eingänge wird das Grafiktablett angeschlossen. Eine A-D-Wandlung ist nicht erforderlich. Jeder Punkt auf dem Grafiktablett hat eine ganz bestimmte x- und y-Koordinate, denen analoge Spannungen zugeordnet werden. Auf den ersten Blick scheint ein Grafiktablett recht einfach zu funktionieren. Um aber die Problematik und die Funktionsweise zu verstehen, muß das Gerät auseinandergenommen werden (Bild 3).

Es lassen sich sehr deutlich zwei Graphitschichten (kontinuierliche Widerstandsschichten) erkennen, von denen eine mit »Pickeln« versehen ist, die in einem quadratischen Raster angeordnet sind. Bei den Pickeln handelt es sich um Isolatoren, die dafür sorgen, daß kein zufälliger Kontakt zwischen den Schichten entsteht. Jede dieser beiden Graphitschichten ist mit zwei Anschlüssen versehen, die genau gegenüber liegen (Bild 4) und so übereinander angeordnet sind, daß die Anschlußpaare einen Winkel von 90 Grad bilden. Eine Schicht repräsentiert die y-Koordinate, die andere die x-Koordinate. Durch einen Druck auf das Tablett werden beide Schichten miteinander verbunden.



▲ **Bild 2. Impulsfolge der beiden Lichtschranken im Trackball.**



◄ **Bild 4. Schematische Darstellung eines Grafiktabletts.**

Kohle als Widerstand

Es bleibt noch die Frage zu klären, wie sich x- und y-Koordinate ausschließlich aus diesem Kontakt ermitteln lassen. Häufig ist zu lesen, daß jedem Punkt auf dem Grafiktablett ein Widerstand zugeordnet ist. Diese Aussage ist aber falsch. Ein Beispiel dazu. Angenommen, wir drücken auf eine beliebige Stelle des Grafiktabletts (gestrichelte Linie in Bild 4) und messen den Widerstand von Klemme 1 nach Klemme 3. Setzt man für x1 und y1 die Zahlenwerte 10- und 20-Einheiten ein, so erhält man als Ergebnis den Summenwiderstand $x1 + y1 = R1 = 10 + 20 = 30$ Einheiten. Ein Druck auf die Stelle $x1 = 20$ Einheiten und $y1 = 10$ Einheiten würde aber zum gleichen Ergebnis führen. Daraus folgt, daß es so nicht geht. Wir haben es hier nämlich mit »einer« Gleichung mit »zwei« Unbekannten zu tun. Eine solche Gleichung ist nicht lösbar. Damit man zu einen eindeutigen Ergebnis kommt, ist noch eine zweite Gleichung (Widerstandsmessung von Klemme 2 nach Klemme 3) erforderlich und zwar $x2 + y1 = R2$. Es ist jetzt noch eine weitere Unbekannte x2 hinzugekommen, die noch eine dritte Gleichung erforderlich macht. Das macht aber nichts, denn der Widerstand einer Graphitschicht, der sich aus $x1 + x2 = Rgr$ zusammensetzt, ist bekannt. Löst man dieses Gleichungssystem nach x1 und y1 auf, so erhält man als Ergebnis:

$$x1 = \frac{1}{2}(R1 - (R2 - Rgr))$$
$$y1 = \frac{1}{2}(R1 + (R2 - Rgr))$$

Elektrotechnisch lassen sich R1 und R2 relativ einfach auswerten. Zu beachten ist allerdings, daß in dem mathematischen Modell die Stromverteilung in den Widerständen als konstant vorausgesetzt wurde. Eine solche konstante Stromverteilung läßt sich durch eine Konstantstromquelle erzielen. Die angeschlossene Meßelektronik, der zur Ermittlung der Werte x1 und y1 die Werte R1 und R2 zur Verfügung steht, sollte einen hohen Eingangswiderstand haben, um die Konstantstromquelle nicht unnötig zu belasten.

Der Lichtgriffel

Genug zum Grafiktablett. Wenden wir uns einem weiteren interessanten Eingabegerät zu, dem Lichtgriffel. Von außen sieht er recht unscheinbar aus und ähnelt in seiner Form eher einem Kugelschreiber. Im Innern jedoch verbirgt sich eine komplizierte Elektronik, die aber bei weitem nicht mit der des Grafiktabletts verglichen werden kann. Außerdem ist ein großer Teil der Elektronik schon im Computer, genauer im Videoprozessor integriert, so daß sich die äußere Elektronik auf ein Minimum beschränkt.

Ein Pin des Videoprozessors Namens LP ist direkt mit dem Controll-Port 1 verbunden, an den der Lichtgriffel angeschlossen werden kann. Lichtgriffel sind genauso wie Trackball oder Grafiktablett Eingabegeräte. Allerdings benötigen sie einen Fernseher oder Monitor, der im Zeilensprungverfahren arbeiten muß (Bild 5). An Monitoren oder Fernsehern, die mit einem anderen Verfahren arbeiten, kann kein Lichtgriffel angeschlossen werden. Dazu ein paar Worte zum Zeilensprungverfahren. Das Fernsehbild, so wie es gesehen wird, existiert nur für das menschliche Auge. In der Tat ist es aber zusammengesetzt aus 625 Zeilen und jede Zeile besteht aus 833 Bildpunkten. Genaugenommen leuchtet immer nur ein Bildpunkt, weil nur ein Kathodenstrahl vorhanden ist, der den Bildpunkt zum Leuchten anregt. Dieser Kathodenstrahl ist aber so schnell, daß er sich in einer Sekunde 15625mal vom linken zum rechten Bildschirmrand bewegt. Um das Bild dem Betrachter sichtbar zu machen, wandert der so erzeugte Bildpunkt innerhalb von 40 ms (0,04 Sekunden) einmal über den gesamten Bildschirm.

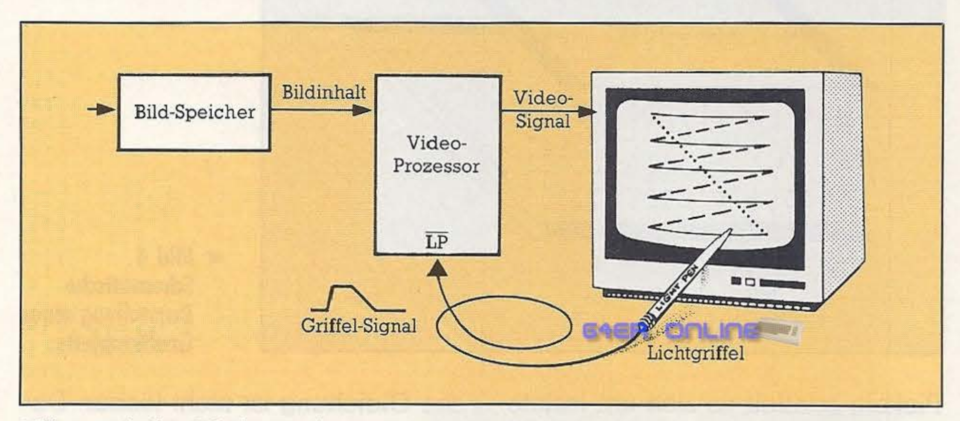


Bild 5. Einfaches Schaltungsschema eines Lichtgriffels

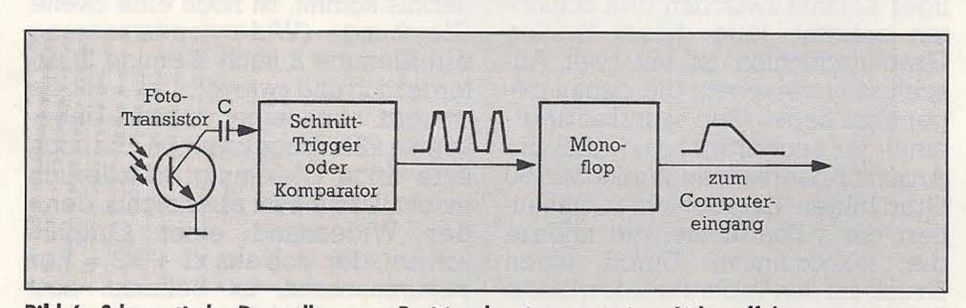


Bild 6. Schematische Darstellung zur Positionsbestimmung eines Lichtgriffels

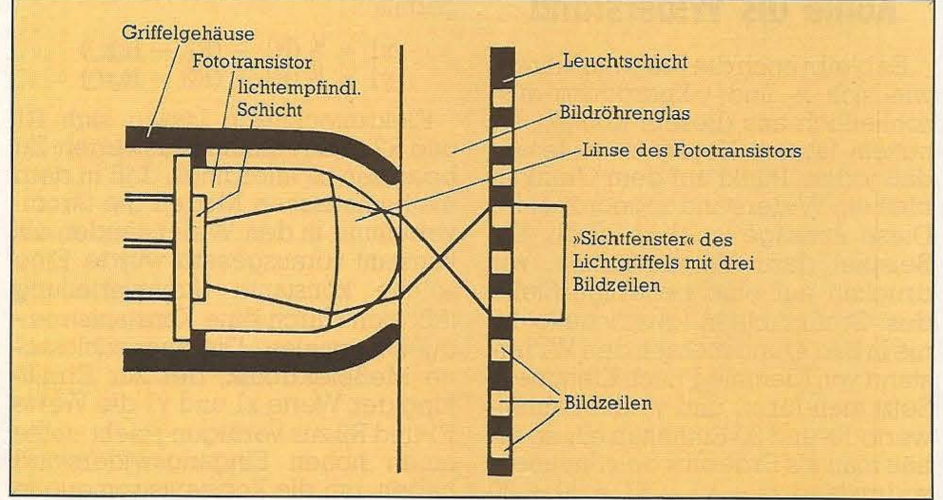


Bild 7. Trotz Linse »sieht« der Lichtgriffel mehr als nur eine Bildschirmzeile

Wird ein Lichtgriffel gegen den Bildschirm gehalten, so registriert ein lichtempflindliches Bauteil im Vorderteil des Griffels diesen vorbeihuschenden Leuchtfleck und sendet ihn als elektrischen Impuls zum Computer. Dieser wiederum weiß, da er den Bildschirm Punkt für Punkt aus dem Bildspeicher ausliest, an welcher Stelle er sich zur Zeit des ankommenden Impulses gerade befindet (Bild 5). Zwar treten zwischen gesendetem und ankommendem Signal Zeitverzögerungen auf, die aber im Videoprozessor kompensiert werden. Als lichtempfindliches Bauteil eignen sich aufgrund der extrem hohen Geschwindigkeit und der niedrigen Leuchtdichte des Kathodenstrahls nur Fototransistoren. Nur ein solches Bauteil ist in der Lage, innerhalb von 76 ns den Leuchtfleck zu erkennen und ausreichend zu verstärken. Um die erforderliche Flankensteilheit zu erreichen, wird das Signal auf einen Schmitt-Trigger geleitet, dessen Ausgang ab einer bestimmten Schaltschwelle schlagartig von Null auf Betriebsspannung springt (Bild 6). Das hat außerdem den Vorteil, daß der Lichtgriffel nicht so allergisch auf Helligkeitsschwankungen des Bildschirms reagiert. Zwischen Kollektor und Eingang des Schmitt-Triggers ist noch ein ausreichend kleiner Kondensator zu legen, der die Aufgabe hat, den Gleichspannungspegel von Lichtquellen wie zum Beispiel der Sonne, auszufiltern und die Wechselspannung, die durch den Impuls des Leuchtfleckes entsteht, durchzulassen.

Da der Fototransistor aufgrund seiner schlechten Optik und des dicken Bildschirmglases (Bild 7) nicht nur einen Punkt, sondern eine ganze Schar von Punkten sieht, ist noch ein weiterer Baustein erforderlich. Ein Mono-Flop hat dafür zu sorgen, daß nach dem ersten Impuls eine gewisse Zeitspanne keine Impulse mehr an den Eingang des Computers gelangen. Die Dauer dieser Zeitspanne darf allerdings die Zeit, die der Kathodenstrahl für das Schreiben eines Bildes benötigt (40 ms) nicht überschreiten. Sonst könnte beim nächsten Durchgang keine neue Positionsbestimmung erfolgen.

(ah)



Malen auf dem Bildschirm

Die verschiedensten Eingabegeräte und die unterschiedlichsten Mal- und Zeichenprogramme kämpfen um die Gunst des Computerkunden. Wir wollen Ihnen mit diesem Vergleichstest die Stärken und Schwächen der einzelnen Malprogramme zeigen.

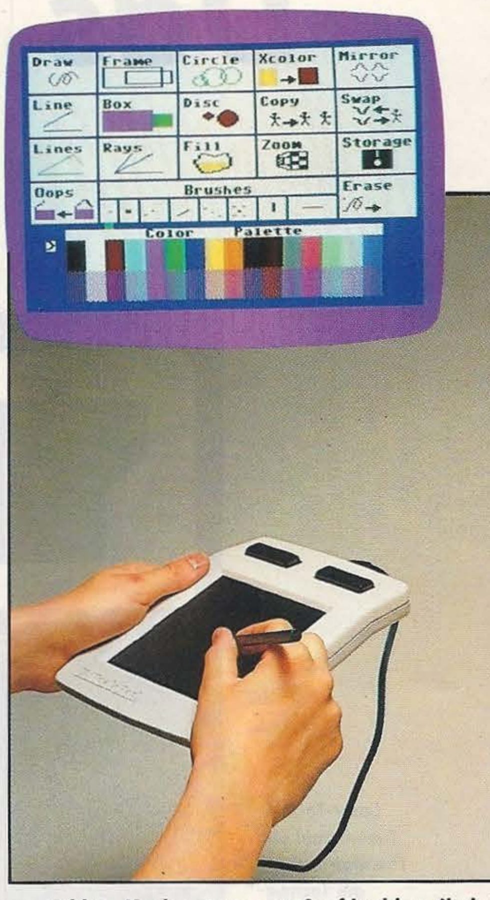


Bild 1. Koalapainter mit Grafiktablett (links)

Möchte man ein Bild nach altbewährter Methode zeichnen, so greift man ganz einfach zu Papier und Bleistift. Auf dem Computer sieht die Sache schon ganz anders aus. Hier sollte man sich zuerst einmal genauer mit der Wahl eines geeigneten Eingabegerätes befassen, denn es gilt, eine erste Entscheidung zu treffen.

Zur Wahl stehen im wesentlichen fünf Eingabearten, die von den jeweiligen Grafikprogrammen unterstützt werden. Da ist zunächst einmal die Tastatur, die ja sowieso bei jedem C 64 vorhanden ist. Wer jedoch wirklich gute Bilder zeichnen will, wird sehr bald feststellen, daß dies mit der Tastatur sehr mühsam ist. Ein anderes billiges Eingabegerät für solche Zwecke ist der Joystick, der auch fast in jedem Computerhaushalt anzutreffen ist. Damit ist es schon eher möglich, schöne Bilder auf dem Computer zu entwerfen.

Wer sich nicht vor größeren Ausgaben scheut, sollte sich ein Grafiktablett zulegen, denn diese Form der Eingabe kommt dem wirklichen Zeichnen am nächsten. Doch auch hier gibt es qualitative Unterschiede zwischen den einzelnen »Brettern« und der dazugehörigen Software, wovon im folgenden noch die Rede sein wird.

Die Software ist im allgemeinen recht gut, doch hier und da sind einige Schwachpunkte zu finden. Fast alle Programme sind mit einem Menübild versehen, mit dem man die einzelnen Funktionen anwählt. Dies kann unter Umständen sehr zeitraubend sein, besonders dann, wenn man mit dem Joystick von einer Ecke in die andere fahren muß. Mit einem Grafiktablett hingegen geht diese Anwahl der einzelnen Menüpunkte wesentlich schneller vonstatten.

Einige Funktionen haben alle Mal- und Zeichenprogramme gemeinsam; sie sind bereits zum Standard geworden und werden daher im folgenden Vergleichstest nur noch beiläufig erwähnt. Dazu gehören zum Beispiel Funktionen wie DRAW (zum freihändigen Zeichnen), PAINT (um geschlossene Flächen zu füllen), BOX (zeichnet ein Rechteck), CIRCLE (zum Zeichnen von Kreisen) und zur Farbauswahl. Auch eine Zoom-Funktion ist in allen Programmen zu finden. Sie erlaubt das Vergrößern einzelner Bildschirmausschnitte, damit man an seinem Werk die nötigen Feinkorrekturen vornehmen kann. Weiterhin ist in fast allen Programmen eine COPY-Funktion integriert. Mit ihr kann man, nach dem Markieren eines Bildschirmteils, dieses frei verschieben und überall hinkopieren. Schon nicht mehr ganz selbstverständlich ist die ABC-Option, die das Einfügen von Buchstaben und Zeichen im zu bearbeitenden Grafikbild erlaubt. Auch ein Drucker, der die Bilder zu Papier bringt, wird nicht von allen Zeichenprogrammen unterstützt.

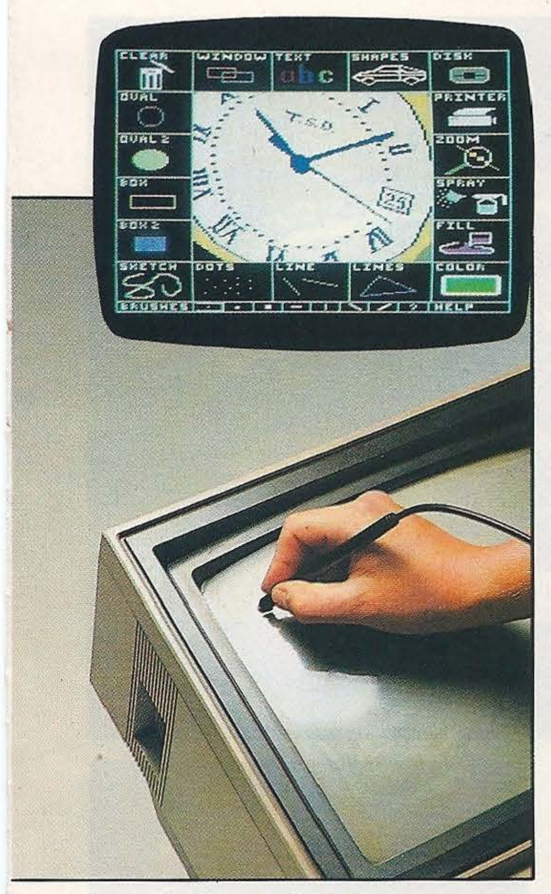
Malen mit dem Koalabär

Einer der ersten Softwarehersteller, die mit einem solchen menügesteuerten Malprogramm auf den Markt kamen, war die Firma KOALA-Technologies. Der Name dieses kleinen australischen Bärs ist zum Synonym für Grafiktabletts geworden. Nichtsdestotrotz ist dieses Gerät — bedingt durch seine Bauart — eines der Besten seiner Art.

Auf einer Kassette befindet sich das eigentliche Zeichenprogramm (Bild 1), das man jedoch der Bequemlichkeit halber lieber gleich auf Diskette abspeichern sollte (die Diskettenoption im Programm unterstützt das Arbeiten mit beiden Massenspeichern). Beim eigentlichen Zeichnen hat man die Wahl zwischen verschiedenen Strichformen und Strichstärken. Diese, wie alle anderen Funktionen, werden über das Menü des Koalapainters angewählt. Dazu fährt man die entsprechende Position an und »klickt« das Feld, mit einer der beiden Tasten auf dem Tablett, an (das Feld beginnt dann zu blinken). Anschließend bewegt man den Cursor aus dem Bildschirmfeld heraus und betätigt nochmals diese Taste. Erst dann schaltet das Programm auf einen der beiden Grafikbildschirme um.

Neben den oben beschriebenen Standardfunktionen finden sich noch weitere interessante Optionen in diesem Menü. Da ist zum Beispiel die Funktion »MIRROR«, die man zusätzlich zu einer Standardoption wählen kann. Damit werden alle nachfolgenden Zeichnungen an zwei imaginären Achsen durch den Bildschirmmittelpunkt gespiegelt, wodurch der Eindruck entsteht, als zeichne man auf dem Bildschirm viermal dasselbe Bild. Eine andere, sehr nützliche Einrichtung ist der Menüpunkt »OOPS«. Mit ihm kann man alle Änderungen am Bild, die nach dem Verlassen des Menüs erfolgten, wieder rückgängig machen.

Sehr komfortabel wurde in diesem Programm auch die Handhabung der Peripheriegeräte gestaltet. Wählt man diesen Menüpunkt (Diskettenzeichen) an, so werden alle Grafikbilder von der sich im Laufwerk befindlichen Diskette aufgelistet.



und Blazing Paddels mit Lichtgriffel

Nachdem man beispielsweise das Feld »GET« angeklickt hat, wird über das Grafiktablett das gewünschte Bild angewählt und vom Programm automatisch geladen. Betrachtet man das Directory der Diskette, so stellt man fest, daß der Koalapainter ein bestimmtes Format des Dateinamens voraussetzt. Damit ist es ihm möglich, Programmfiles von Grafikbildern zu unterscheiden. Bedingt durch dieses eigenwillige Dateinamenformat, lassen sich Bilder von anderen Zeichenprogrammen nicht ohne Änderung des Bildnamens laden.

Koalapainter in zwei Versionen

Fast genau das gleiche Programm liegt der Lichtgriffelversion des Koalapainters bei. Diese wurde lediglich um die bereits oben erwähnte ABC-Funktion erweitert, so daß man auch Buchstaben und Zeichen mit in seine Kunstwerke übernehmen kann.

Gearbeitet wird mit dem Griffel fast so wie mit dem Tablett. Nachdem man einen Menüpunkt angefahren hat, drückt man mit dem Stift gegen den Schirm, um das Feld anzuwählen. Die Spitze des Lichtgriffels ist nämlich gleichzeitig ein Schalter, den man durch Drücken gegen den Bildschirm schließt. Leider hat genau dieses bei unserem Testgerät nicht funktioniert.

Überhaupt erzielt man meines Erachtens mit diesen Stiften nicht so gute Ergebnisse wie mit dem Grafiktablett, denn gerade bei größeren Zeichnungen ermüdet der Arm, der den Griffel ja immer am Bildschirm halten muß, sehr schnell. Abhilfe könnte hier höchstens das Drehen des Schirmes (mit der Bildfläche nach oben) schaffen.

Wenn man dann nach langer Arbeit sein Kunstwerk (Bild 2) gezeichnet und abgespeichert hat, möchte man es vielleicht auch einmal zu Papier bringen. Leider sucht man die Druckeroption in den Koalapaintermenüs vergebens, denn eine solche Funktion wurde nicht implementiert. Das nötige Druckprogramm muß gesondert gekauft werden. Damit werden alle Grafiken wirklichkeitsgetreu ausgegeben, denn das Programm verwandelt alle 16 Farben in 16 Grauwerte (Bild 3). Ein Unterprogramm ermöglicht sogar für die einzelnen Farben die Definition dieser 16 Druckmuster. Unterstützt werden die gängigsten Drucker für den C 64, wie zum Beispiel die Epson-Printer und die Commodore-Drucker.

Die Bedienung dieses Druckerprogramms erfolgt wiederum mittels Koalapad, der Lichtgriffel wird allerdings nicht unterstützt. So können die Besitzer des Koala-Lightpens momentan noch nichts mit diesem Programm anfangen, denn eine Anwahl der einzelnen Menüpunkte ist ohne Brett nicht möglich!

Malen mit dem Finger

»Touch Point« ist der Name eines andern Grafiktabletts, das ebenfalls getestet wurde. Dieses, im Vergleich mit dem Koalapad fast doppelt so großes Gerät, wird in den Joystick-Port 1 eingesteckt. Aufgrund seiner Größe bietet es sich an, die Zeichnungen mit dem Finger zu erstellen. An dem Brett befinden sich auf jeder Seite zwei Taster, so daß auch Linkshänder beim Zeichnen auf der — mit einem Millimeterraster verzierten — Zeichenfläche, keine Probleme haben.

Weiterhin liegen noch ein kleines Bedienerhandbuch und ein Steckmodul diesem Tablett bei. Die dazugehörige Software befindet sich also in einem Modul und muß daher nicht immer wieder geladen werden. Nach dem Einschalten erscheint ein Menübild, dessen Inhalt dem des Koalapainters sehr ähnlich ist. Einige Funktionen, wie zum Beispiel der Menüpunkt »GRID«, sind neu hinzugekommen. Mit »GRID« erzeugt man ein Gittermuster in dem aktuellen Grafikbild, was gerade das Zentrieren der Bilder vereinfacht. Auch Dreiecke und Pyramiden können auf einfache Weise gezeichnet werden. Sogar die Ansteuerung eines Druckers per Menüanwahl ist möglich. Mit welchen Geräten dieses Programm zusammenarbeitet, kann man dem beigelegten Handbuch leider nicht entnehmen.

Auch die »DRAW«-Funktion weist eine große Schwäche auf. Mir ist es während des gesamten Tests nicht gelungen, eine durchgehende Linie zu ziehen. So riß die Linie immer wieder auf und »zerlief« in einzelne Punkte. Dieses Manko ist jedoch nicht der billigen Bauart des Grafiktabletts (statt einer kontinuierlichen Widerstandsschicht, wie beim Koalapad, wurde hier eine Drahtmatrix unter der Malfläche aufgespannt) anzulasten, sondern der Zeichensoftware. Denn auch mit dem qualitativ besseren Koalapad und diesem Programm ist es uns nicht gelungen, eine durchgehende Linie zu ziehen.

Blazing Paddels — ein Malprogramm für alle Anwendungen

Ein Grafikprogramm mit ganz besonderen Fähigkeiten ist BLAZING PADDELS von Baudville. Es wird ebenfalls durch ein Menü (Bild 1) bedient und unterstützt fast alle gängigen Eingabegeräte. Die Auswahl ist direkt nach dem Programmstart zu treffen und kann dann während des Laufes nicht mehr revidiert werden.

Neben den bekannten Standard-Optionen sind in diesem Programm einige weitere interessante Optionen integriert worden. So ist es zum Beispiel möglich mit 200 verschiedenen Farben zu zeichnen, was durch einen Programmiertrick ermöglicht wird. Da der C 64 natürlich nicht in der Lage ist, so viele verschiedene Farben darzustellen, werden von dem Programm zwei verschiedenfarbige Multicolor-Pixels so aneinander gesetzt, daß der Eindruck einer neuen Farbe entsteht.

Nicht minder interessant ist der Menüpunkt »SHAPES«. Shapes sind hier vordefinierte Grafiksymbole, die man nach Belieben aufrufen, drehen, spiegeln und anschließend ins aktuelle Bild einbauen kann. Da-

durch muß man nicht mehr wirklich jedes Detail des Bildes selbst zeichnen. Auf der Programmdiskette befinden sich einige Files, von dem eins jeweils 20 Symbole beinhaltet. Leider kommt man mit diesen vorgefertigten Standardzeichen nicht sehr weit. Daher bietet die Firma Baudville inzwischen zwei Zusatzdisketten an. Selbst kann man diese Shapes jedoch nicht definieren, so daß man in diesem Punkt auf die nicht ganz billigen Erweiterungen angewiesen ist.

Zwei weitere Besonderheiten dieses Zeichenprogramms sind die Window-Funktion und die ladbaren Zeichensätze. Auch die einzelnen Zeichen im Text-Modus müssen von Disk nachgeladen werden. Der Vorteil ist, daß viele verschiedene Schriftarten in ein und demselben Bild dargestellt werden können. Aber auch hier gilt das gleiche wie bei den Shapes: Einige Zeichensätze werden mitgeliefert, die anderen befinden sich auf den Erweiterungsdisketten; selbst editieren ist auch hier nicht möglich. Dafür kann man seine Grafikfenster nach Belieben abspeichern und laden.

Ein Fenster entsteht ganz einfach durch das Markieren eines bestimmten Zeichenbereiches in dem Grafikbild. Dazu wählt man den entsprechenden Menüpunkt an, »zieht« ein Rechteck und markiert damit den gewünschten Bereich, der dann zunächst zwischengespeichert wird. Diesen kann man dann aber auch auf Diskette abspeichern.

Ähnlich wie beim Koalapainter markiert auch dieses Programm seine Files durch Zusätze im Dateinamen, um sie voneinander unterscheiden zu können. Bilder erhalten zum Beispiel den Vorsatz »PI.« zum eigentlichen Namen, die Windows werden mit dem Suffix »WI.« abgespeichert.

Auch das Drucken der grafischen Wunderwerke ist mit diesem Programm möglich, denn hier wurde wiederum eine Druckeroption eingebaut. Man hat sogar die Wahl zwischen unterschiedlichen Druckern (Epson, Commodore und Gemini).

Abschließend kann man sagen, daß Blazing Paddels das beste Zeichenprogramm ist. Dieses Urteil verdankt es vor allem seinen umfangreichen Funktionen, von denen im Rahmen dieses Tests nur einige herausgestellt werden konnten.

Ein anderes, sehr farbiges Mal- und Zeichenprogramm ist Magic-Paintbrush von Pingouin-Soft. Es ist, genauso wie Blazing Paddels, in der Lage 255 verschiedene Mischfarben darzustellen. Daneben erlaubt es den Anschluß von zwei unterschiedlichen Eingabegeräten (Joystick und Grafiktablett), die parallel, also nebeneinander, benutzt werden können. So ist es möglich, die guten Eigenschaften beider Geräte zu nutzen; also zum Beispiel das Grafiktablett zum Zeichnen und den Joystick für die Feinarbeit zu verwenden. Aber auch die Tastatur kann zum Zeichnen und Bedienen herangezogen werden.

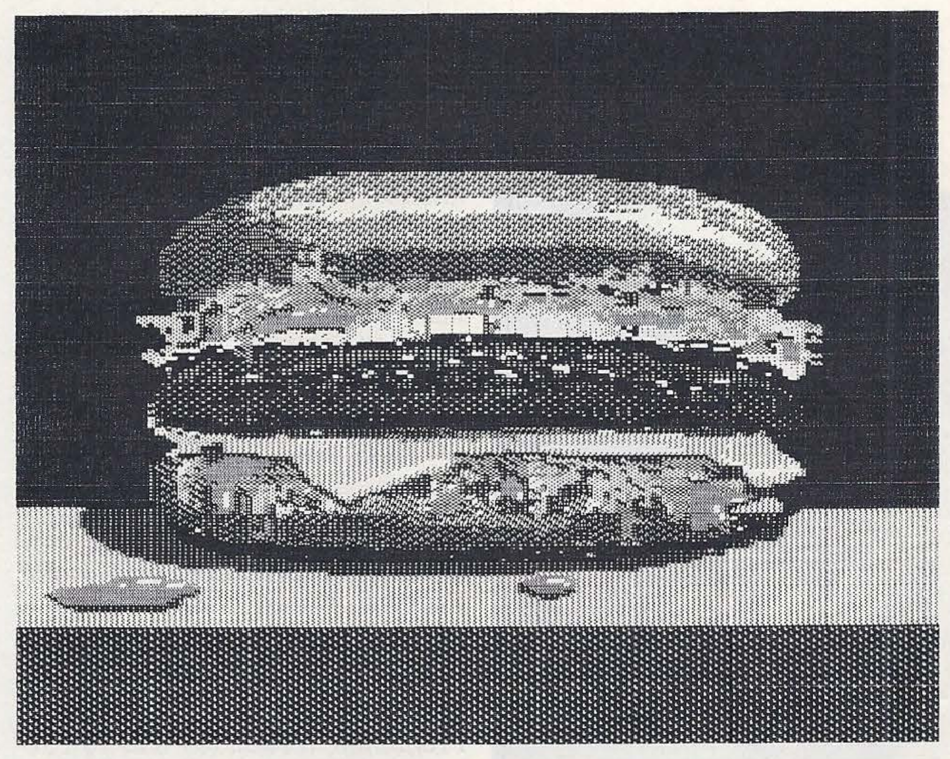


Bild 3. Die erzeugte Hardcopy

Magischer Pinselstrich

Weitere Glanzpunkte dieses Programms sind neben der Farbigkeit vor allem die verschiedenen Pinselformen. Normalerweise erscheint beim Zeichnen unter dem Cursor (hier ist es ein Fadenkreuz) ein kleiner Punkt. Viele dieser Punkte ergeben logischerweise eine Linie oder zumindest einen Strich. Fast alle Malprogramme erlauben die Benutzung verschiedener Pinselformen, so daß statt dem kleinen Punkt ein größerer Punkt oder ein kleines Muster erscheint. Magic Paintbrush stellt dem Benutzer sage und schreibe 100 verschiedene Pinselmuster zur Verfügung (daher auch der Name). Leider ist der Zeichenpinsel in diesem Modus sehr langsam, so daß man für diese Funktion viel Geduld mitbringen muß.

Ansonsten sind die üblichen Standardfunktionen, wie zum Beispiel Draw, Line, Box und Circle, in diesem Programm zu finden; eine Druckeroption ist leider nicht vorhanden.

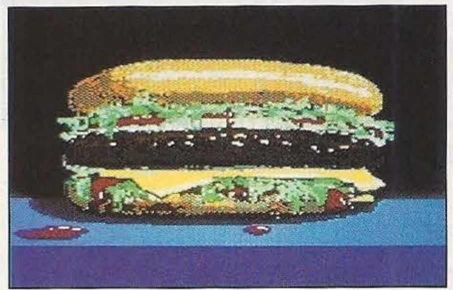


Bild 2. Hamburger mit Koalapainter

Das preiswerteste Programm

Das Zeichenprogramm »Paint Magic« von Markt und Technik geht bezüglich der Bedienung einen etwas anderen Weg. Hier findet man nach dem Laden kein Menübild, sondern nur eine Übersichtstafel, die über die wichtigsten Programmfunktionen informiert. Diese werden direkt, also mittels Tastendruck angewählt. Das hat den Vorteil, daß man nicht jedesmal in dem Menübild hin- und herfahren braucht, um die gewünschten Funktionen zu selektieren. Dafür benötigt man jedoch eine gewisse Einarbeitungszeit, um sich mit den einzelnen Programmpunkten vertraut zu machen.

Neben den üblichen Möglichkeiten bietet dieses Programm, das nur den Joystick als Zeichengerät unterstützt, dem Zeichner gleich zwei Bildschirmseiten für seine Grafiken an. Dies kann sonst nur der Koalapainter, die Touchpoint Software und Hi-Eddi (sogar sieben Seiten).

Eine weitere Besonderheit von Paint Magic ist das Pinselmenü. Mit ihm können vier verschiedene Schraffurmuster festgelegt werden, die dann später beim Füllen geschlossener Flächen Verwendung finden können.

Da sich der Cursor bei manchen Funktionen relativ langsam bewegt, kann man die Geschwindigkeit mit den Tasten 1-8 auf den für sich idealen Wert einstellen. Gerade beim Zeichnen von Kreisen und Rechtecken wirkt sich diese Option positiv aus.

Übrigens ist dieses Programm nur auf Diskette zu haben, was die Benutzer mit Datasette nicht sehr erfreuen dürfte. Neben dem eigentlichen Programm werden auf der Paint-Magic-Diskette noch acht fertige Beispielbilder, die recht schön sind, mitgeliefert.

Die Schnecke

Recht gemächlich geht es in dem Programm »Paint Pic« von Data Becker zu. Auch hier wurde auf ein aufwendiges Menübild verzichtet, denn die Anwahl der Funktionen geschieht wieder über einzelne Tasten. Hat man dann zum Beispiel die »DRAW«-Funktion angewählt, erlebt man eine böse Überraschung. Die Geschwindigkeit des Zeichenstiftes ist alles andere als schnell. Auch beim Anfertigen des Bildes störte immer wieder, daß alles nur äußerst langsam vonstatten geht. So ist es zwar möglich, mit der HOME-Taste neun verschiedene Bildschirmpositionen anzufahren, aber eine Taste zur Beschleunigung des Cursors war nicht zu finden. Noch schlimmer wirkt sich die Geschwindigkeit bei der Circle-Funktion aus. Gerade bei großen Kreisen muß man oftmals eine halbe Minute (!) warten bis Paint Pic mit dem Zeichen fertig ist. Da liegt dann doch der Verdacht sehr nahe, daß das gesamte Programm in Basic geschrieben worden ist.

Dieses große Handicap überschattet alle anderen positiven Eindrücke von diesem Zeichenprogramm. Da sind zum Beispiel mehrere Funktionen zum perspektivischen Zeichnen integriert, mit denen man in der Lage ist, auf einfache Art und Weise räumliche Bilder zu entwerfen. Auch das sehr ausführliche, didaktisch aufgebaute Handbuch sollte an dieser Stelle einmal erwähnt werden.

Dennoch kann ich dieses Programm nicht ohne Vorbehalte weiterempfehlen, da gerade ein Zeichenprogramm einigermaßen schnell sein sollte.

Hi-Eddi und die »Profis«

Daß abgetippte Programme den Vergleich mit käuflichen nicht scheuen brauchen, zeigt das Programm Hi-Eddi (Ausgabe 1/85). In vielen Punkten übertrifft es so manches »professionelle« Zeichenprogramm.

Hi-Eddi weist außergewöhnliche Leistungsmerkmale auf, was bereits an den Menübildern zu erkennen ist. Die Steuerung erfolgt mittels Joystick, die Befehlseingabe kann aber auch über die Tastatur erfolgen.

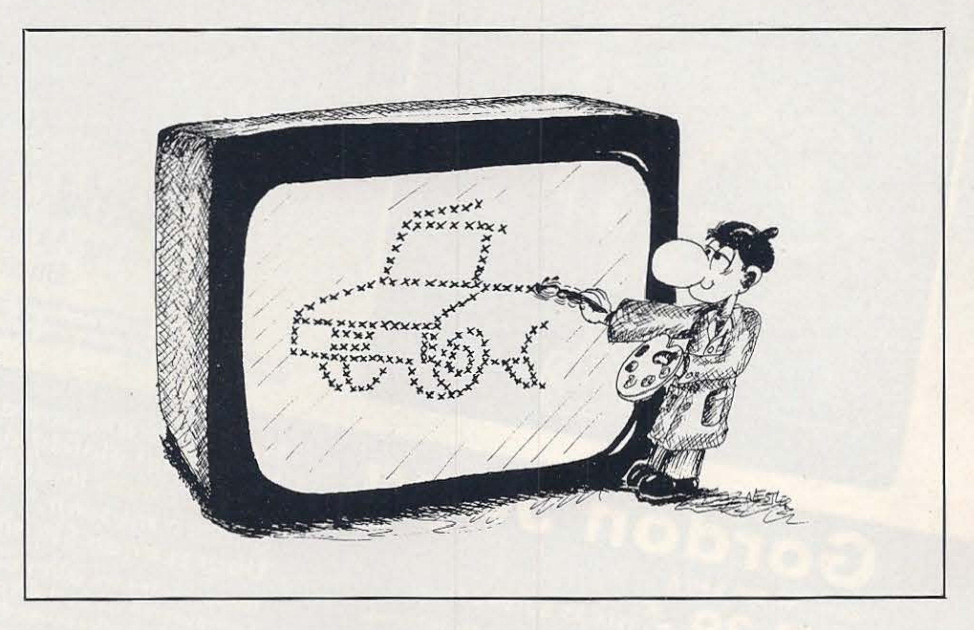
Bereits der Cursor ist eine angenehme Überraschung. Er beschleunigt nach einiger Zeit seinen Lauf, damit auch größere Distanzen schnell überwunden werden können. Neben den Standardoptionen bietet Hi-Eddi eine ganze Reihe von Sonderfunktionen. Da ist zum Beispiel das Menüfeld »SPRITE«, das den Spriteeditor anwählt. Damit ist es möglich, ein Sprite in vergrößerter Darstellung zu bearbeiten. Mit den Befehlen »STAMP« beziehungsweise »APPEND« kann man das Sprite wieder in den Grafikbildschirm »einpflanzen«, wobei im ersten Fall der Hintergrund gelöscht wird und im zweiten nicht. Auch das Heraustrennen eines Bildschirmflickens ist möglich; mit »GET« übernimmt man eine spritegroße Fläche in den Editor.

Bemerkenswert ist auch die Tatsache, daß Hi-Eddi sechs — im Schwarzweißmodus sieben — Bildschirmseiten verwalten kann, womit sich dem Zeichner völlig neue Perspektiven eröffnen. So kann man zum Beispiel mittels eines einfachen Befehles den Inhalt eines Bildschirms in einen anderen kopieren. Damit ist es auch möglich, auf einfache Weise Bewegungsphasen zu programmieren, indem man einmal ein Bild entwirft, es in den nächsten Bildschirm kopiert, dort abändert, wieder kopiert und so weiter. Der »WALK«-Befehl schaltet dann einfach die einzelnen Seiten durch. Auf diese Weise entsteht ein echter Zeichentrickfilmeffekt.

Hi-Eddi unterstützt auch angeschlossene Drucker. Die in Ausgabe 1/85 veröffentlichte Druckroutine erlaubt nur die Ansteuerung eines Epson-Druckers. Inzwischen ist dieses Unterprogramm auch für andere Drucker umgeschrieben und auch bereits veröffentlicht worden (für MPS 802/801).

Zusammenfassend kann man sagen, daß jedes Programm, abgesehen von Paint Pic, zu empfehlen ist. Ausschlaggebendes Kriterium dürfte daher letztendlich nur noch der Preis für die einzelnen Programme sein. Da ist natürlich das Preis-/Leistungsverhältnis von Hi-Eddi am günstigsten. Aber auch Paint Magic schneidet mit einem Preis von 59 Mark recht gut ab. Wesentlich mehr muß man hingegen für Blazing Paddels hinblättern; 169 Mark kostet das Grundprogramm (79 Mark die Erweiterungen). In derselben Preisklasse liegt dann auch das Touch Point (149 Mark inklusive Modul) und der Koala-Lightpen (ca. 200 Mark). Mit Abstand am teuersten ist jedoch das Koalapad, das für 298 Mark zu haben ist. (Christoph Sauer/ah)

Bezugsquellen:
Koalapad: Harman Deutschland, Hünderstr. 1, 7100 Heilbronn
Touch Point: Ce-tec, Lange Reihe 29, 2000 Hamburg 1
Blazing Paddels und Magic Paintbrush: Softline, Schwarzwaldstr. 8A, 7602 Oberkirch
Paint Magic: Happy Software, Markt & Technik Verlag AG, Hans-Pinsel-Str. 2, 8013 Haar
Paint Pic: Data Becker, Merowingerstr. 30, 4000 Düsseldorf
Hi-Eddi: Listing des Monats, 64'er, Ausgabe 1/85



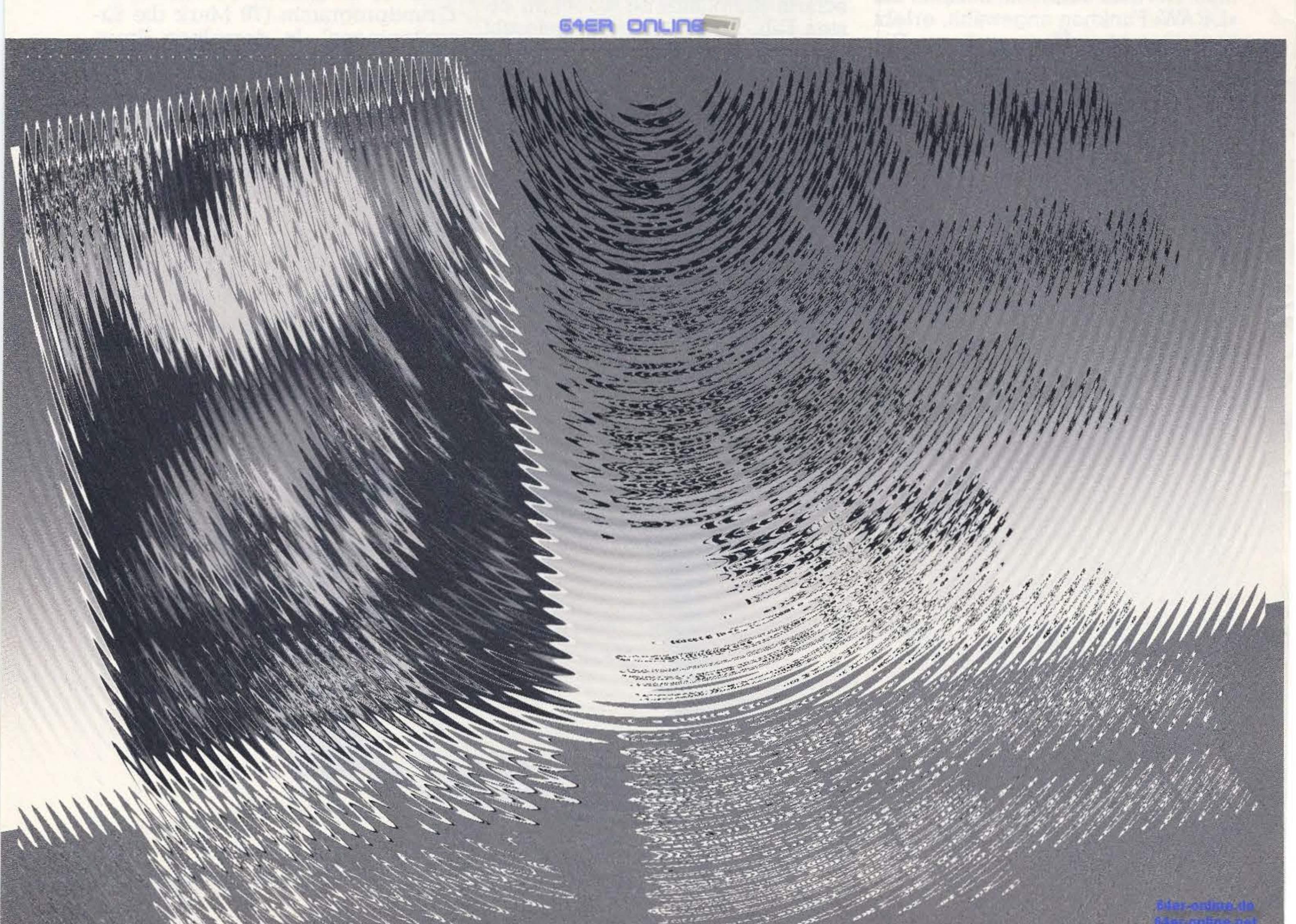
Grafikprogramme auf einen Blick

Sie möchten die Grafikfähigkeiten des C 64 nutzen? Dann brauchen Sie Grafiksoftware. Diese Marktübersicht soll Ihnen die Auswahl erleichtern.

Wer sich an seinen C 64 setzt und versucht Grafiken auf den Bildschirm zu zaubern, wird enttäuscht sein. Zum einen fehlen im Basic-V2.0 jegliche Grafikbefehle, zum anderen ist die Programmierung des Grafikprozessors über PEEK- und POKE-Befehle sehr kompliziert und extrem langsam. Aus diesem Grund haben viele Firmen, kurz nachdem der C 64 auf den Markt kam, Basic-Erweiterungen zu diesem Thema entwickelt. Diese Programme sind laufend verbessert worden und haben heute einen Stand erreicht, der kaum noch Wünsche offen läßt. Mit ihnen können die Grafikmöglichkeiten, die nun einmal in diesem Computer stecken, bis ins letzte vom Basic aus genutzt werden. Aber damit noch nicht genug. Auch wurden Routinen in diese Erweiterungen integriert, die das Zeichnen von Kreisen, Rechtecken und allen möglichen anderen geometrischen Figuren erleichtern. Mit einfachen Basic-Befehlen können diese Routinen in eigene Programme eingebaut werden. So gehört zum Beispiel die Window-Technik und das Arbeiten mit mehreren Bildschirmseiten zum Standard vieler Programme dieser Art.

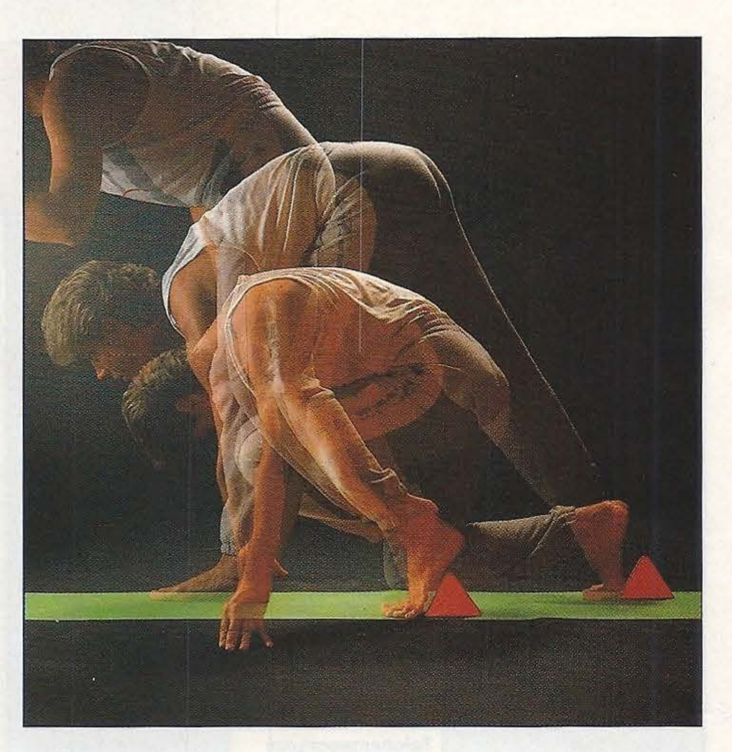
Es gibt aber nicht nur grafikunterstützende Basic-Erweiterungen, die dem Programmierer das Arbeiten mit dem Computer erleichtern. Die zweite Gruppe der Grafiksoftware sind die Zeichenprogramme. Im Gegensatz zu den Programmierhilfen handelt es sich bei diesen Programmen um reine Anwendersoftware. Hier wird nicht mehr programmiert, sondern gemalt. Sämtliche Grafikbefehle können entweder über ein Menü oder über die Tastatur angewählt werden. Diese Art der Grafiksoftware eignet sich besonders gut für diejenigen, die daran interessiert sind, schnell und problemlos Bilder auf den Bildschirm zu bekommen. Die Betonung liegt auf Bilder, denn Funktionskurven oder Grafiken, wie sie mit Programmierhilfen erstellt werden können, entfallen bei dieser Programmart. Dafür können aber die so entstandenen Bilder in eigenen Programmen weiterverarbeitet werden, vorausgesetzt man findet einen Weg diese »Hi-Resolution-Bildschirme«, die immerhin einen Speicherbedarf von 8 KByte haben, in Basic-Programme einzubinden.

In dieser Marktübersicht werden nur die beiden zuvor beschriebenen Softwaregruppen aufgeführt. Alle Angaben beziehen sich auf Auskünfte der Hersteller/Anbieter. (ah)



a) Produkt b) Hersteller c) Anbieter	Typ der Grafik-software	a) Datenträger b) Eingabegerät c) Begrenzung der Farben?	a) Preis mit MwSt. b) Preis inkl. Eingabegerät? Welches?	Funktionen, Besonderheiten
a) **Paint Magic** b) Datamost c) Happy Software, Markt&Technik Verlag AG, Hans-Pinsel-Str. 2, 8013 Haar	Zeichenprogramm	a) Diskette b) Joystick c) 5 Farben und Mischfarben	a) 59 Mark b) nein	Linie, Rechteck, Kreis, Punkt, Strahlen, Füllen: durchgehend, horizontal, vertikal, diagonal, Geschwindigkeit, Muster, Überlagern, Kopieren, Vergrößern, Verkleinern, zweite Leinwand
a) **Blazing Paddles** b) Baudville c) Softlinie, Schwarzwaldstr. 8A, 7602 Oberkirch	Zeichenprogramm	a) Diskette b) Joystick, Grafik-Tablett, Maus, Lichtgriffel, Trackball, Paddles c) nein	a) 169 Mark b) nein	Kreis, Ellipse, Rechteck (leer und ausgefüllt), Sketch, Dots, Linie, Lines, Color, Fill, Spray, Zoom, Printer, Disk, Shapes, Text, Window
a) **Paint-Pic** b) Data Becker c) Data Becker, Merowingerstr. 30, 4000 Düsseldorf	Zeichenprogramm	a) Diskette b) Joystick, Tastatur c) 4 Farben	a) 99 Mark b) nein	Kreis, Rechteck, Parallelogramm, Spiegeln, Drehen von Objekten, Textmodus, Dreieck, Linie
a) **Koala Lightpen** b) Koala Technologies c) Harman Deutschland, Hünderstr. 1, 7100 Heilbronn	Zeichenprogramm	a) Diskette b) Lichtgriffel c) nein	a) zirka 200 Mark b) ja, Lichtgriffel	Box, Circle, Draw, Line, Lines, Copy, Color, Align, 8 verschiedene Schriftarten, Programm für Dia-Show, Mirror
a) **Koala Pad** b) Koala Technologies c) Harmann Deutschland	Zeichenprogramm	a) Diskette, Kassette b) Grafik-Tablett c) nein	a) zirka 298 Mark b) ja, Grafik-Tablett	Draw, Frame, Circle, X-Color, Mirror, Line, Box, Disk, Copy, Swap, Lines, Rays, Fill, Zoom, Storage, Oops, Brushes, Erase, Patterns
a) **Super Sketch** b) Personal Peripheries c) Rushware, An der Gümpgebrücke 24, 4044 Kaarst 2	Zeichenprogramm	a) Diskette, Kassette, Modul b) Grafik-Tablett c) k.A.	a) ab zirka 248 Mark b) ja, Grafik-Tablett	Clear, Draw, Swap, Fill, Erase, Undo, Page, Brush, Design, Lines, Rays, Box, Circle, Window, Copy, Mirror, Quad, Flip, Text, Show, Zoom, File, Printer-Utility, Zusatz-Software (Musik, Architektur, Business)
a) **Tech-Sketch** b) Tech-Sketch USA c) Rushware	Zeichenprogramm	a) Diskette, Kassette b) Lichtgriffel c) nein	a) zirka 149 Mark b) ja, Lichtgriffel	Draw, Point, Line, Lines, Rays, Fill, Frame, Box, Circle, Disk, Erase, Mirror, Magnify, Color, Brush, File, Drucker-Routine in der Software enthalten
a) **Touch-Point** b) Video technology, Hongkong c) Ce-tec, Lange Reihe 29, 2000 Hamburg 1	Zeichenprogramm	a) Modul b) Grafik-Tablett c) k. A.	a) 149 Mark b) ja, Grafik-Tablett	Grid, Draw, Fill, Correct, Line, Lines, Rays, ExColor, Frame, Block, Enlarge, Swap, Triange, Pyrimid, Copy, Storage, Rings, Disc, Symmetry, Clear, Brush Strokes
a) **Hi-Eddi** b) Leser Service (1/85) c) Markt&Technik Verlag AG	Zeichenprogramm	a) Diskette b) Joystick c) Pro 8x8 Punktfeld, eine Farbe	a) 29,90 Mark b) nein	Draw, Line, Rectangle, Circle, Paint, Move, Text, Get, Append, Stamp, Erase, (Invertieren, UND, ODER, EXoder über 7 Bildschirmseiten) Sprite-Editor, Mirror, Rotate, Grid, Print
a) **Supergrafik 64** b) Data Becker c) Data Becker	Programmierhilfe	a) Diskette b) Tastatur c) 4 Farben	a) 99 Mark b) nein	Ellipse, Kreis, Rahmen, 16 Sprites gleichzeitig darstellbar, Ton-Befehle, Utilities wie »Renum«, »Merge«
a) **Leonardo** b) Creative Sparks c) Thorn Emi, Mathias-Brüggen-Str. 21, 5 Köln 30	Programmierhilfe	a) Kassette b) Tastatur c) k. A.	a) zirka 39 Mark b) nein	Über 40 Befehle
a) **Designer's Pencil** b) Activision c) Supersoft, H. Stein, Hohefeldstr. 55, 1 Berlin 28	Programmierhilfe	a) Kassette b) Joystick, Tastatur c) 4 Farben	a) 60 Mark b) nein	Mischform aus Basic und Logo, zusätzlich Fill, Kaleidoskop, Circle, Speed, Color, Recurse, Write, Musik-Befehle
a) **Graff** b) Profisoft c) Rushware	Programmierhilfe	a) Diskette b) Tastatur c) k. A.	a) zirka 69 Mark b) nein	Text, Clear, Color, Plot, Point, Line, Box, Chart, Radius, Circle, Area, Fill, Draw, Sprite, File
a) **Screen Graphics** b) Abacus Software c) k. A.	Programmierhilfe	a) Diskette, Kassette b) Tastatur c) nein	a) k.A. b) k.A.	Multi, Tic, Rot, Draw, Box Circle, CHAR, Block, Fill, Pixel, Dump, Gread, Norm, Graph, Copy Sprite, Off, Place, Bit, Colors, Hex, Gdata
a) **Grafik 2000** b) Leser Service (SH 4) c) Markt&Technik Verlag AG	Programmierhilfe	a) Diskette b) Tastatur c) nein	a) 29,90 Mark b) nein	Clear, Color, Change, Invers, Comp, Gsave, Gload, Point, Hmark, Vmark, Hline, Vline, Line, Circle, Ellipse, Text, Fill, Duplicate, Scroll, Window, Lowcol, Sprite, Ssave, Sload, Screen, Plot, Test
a) **Graphics Basic** b) HesWare c) Ariola-Eurodisc GmbH, Postfach 1350, 4830 Gütersloh 1	Programmierhilfe	a) Diskette b) Tastatur c) nein	a) 89 Mark b) nein	Multi, Hot Line, Box, Circle, Fill, Scale, Text, Window, Sprite Editor, Sprite animation Print, Copy, Sprite
a) **Extended Graphic Systems** b) Interface Age c) Interface Age, Josephsburgstr. 6, 8 München 80	Programmierhilfe Zeichenprogramm	a) Diskette b) Tastatur c) nein	a) 138 Mark b) k.A.	Bit map, Sclear, Pencolor, Move, Draw, Dot, Frame, Circle, Write, Gsave, Gload, Sbase, Sprite, Sglobal, Smulti, Smove, Sground, Expand, On collision goto, Sreturn

Sprites ohne Geheimnisse



Sprites sind für manchen Einsteiger reizvolle, aber schwer zu programmierende Gebilde. Man muß viele Register kennen und mit Binärzahlen arbeiten. Dieser Artikel soll Ihnen helfen, beide Hürden zu meistern.

In Bild 1 finden Sie einen Fahrplan, der in 13 Stationen den Weg zum funktionierenden Sprite enthält. Diesem Schema werden wir nun folgen und anhand von zwei Sprites Abschnitt für Abschnitt ein Sprite-Programm aufbauen.

Schritt 1

Der erste Schritt spielt sich im Kopf ab: Wir fällen eine Entscheidung darüber, ob unser Sprite einfarbig (dafür aber feiner strukturiert) oder mehrfarbig werden soll. Von dieser Entscheidung hängt nämlich Schritt 2 ab. Wir werden zweispurig weiterfahren, indem wir sowohl ein einfarbiges als auch ein »Multicolor«-Sprite wählen.

Schritt 2

Hier nehmen wir uns nun ein Blatt kariertes Papier zur Hand und zeichnen darauf ein Rechteck mit 24 Karos horizontaler und 21 Karos vertikaler Ausdehnung. Das wird unser Sprite-Raster. Dann warten wir auf den Kuß der Musen, der uns eine geeignete Sprite-Gestalt eingibt. Sind wir soweit, dann trennen sich die Wege:

a) Das einfarbige Sprite tragen wir in unser Schema ein wie ein Stickmuster: Überall, wo ein Bildpunkt erscheinen soll, machen wir ein Kreuzchen (eine 1). Dann ergibt sich unser erstes Sprite wie Bild 2 zeigt.

b) Etwas komplexer ist das mit dem Multicolorsprite. Hier treten vier Farben auf, die sich durch die Zahlenkombinationen 00, 01, 10 und 11 unterscheiden lassen. 00 besitzt die Hintergrundfarbe (ist also durchsichtig). Am besten benutzt man drei verschiedene Farbstifte (je einen für 01, 10 und 11) und trägt in das Schema mit dem jeweiligen Stift die Zweierkombination ein. Das Ergebnis sieht dann so aus wie in Bild 3.

Schritt 3

Nach diesen — mehr schöpferischen — Tätigkeiten kommt nun die Mathematik zu Wort. Wir müssen nämlich nun das, was wir auf dem Papier vor uns haben, in eine dem Computer verständliche Form bringen, also in Zahlen. Sehen Sie sich dazu nochmals Bild 2 und 3 an. Dort wurden zwei senkrechte Linien gezogen, die jede Zeile in drei Achtergruppen aufteilt. Eine solche Gruppe ist der künftige Inhalt eines Bytes. Jedes einzelne Kästchen entspricht einem Bit, und schon sind wir auf diese Weise bei den Binärzahlen gelandet. Wir werden aber hier keine Erklärung über alle Höhen und Tiefen dieser Zahlenart starten, sondern nur feststellen, auf welche Weise wir das Gebilde, das wir vor uns haben, in eine normale Zahl umrechnen können. Das ist ganz einfach. Sehen Sie sich dazu Bild 4 an.

Sie finden darin ein Byte und die Numerierung der Bits. Überall dort, wo an einer Bit-Position eine 1 steht, merkt man sich die im Bild darunter stehende Zahl. Alle auf diese Weise gemerkten Zahlen addiert man schließlich. Das Ergebnis ist die Dezimalzahl, die uns interessiert. Ein Beispiel soll das noch verdeutlichen. In Bild 3 (dem Multicolorsprite) finden Sie in Zeile 2 als erstes Byte: 10100000. Wenden wir nun unser Bild 4 darauf an:

1 0 1 0 0 0 0 0
128 64 32 16 8 4 2 1

Sie sehen, wir müssen addieren:

128 + 32 = 160

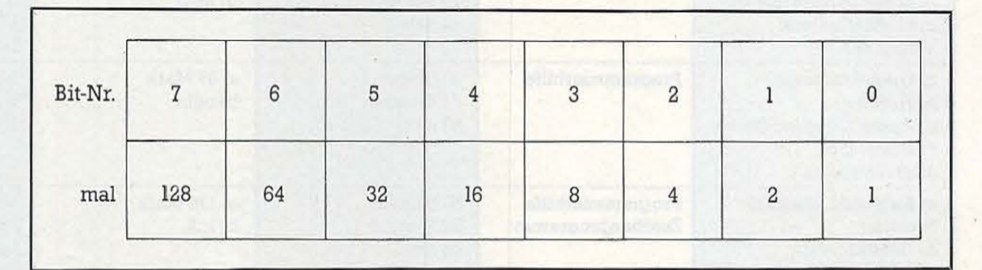
Nun wissen Sie, wie es gemacht wird. Üben können Sie das nun bei allen 126 Byte unserer beiden Sprites. Was dabei herauskommt, sehen Sie ebenfalls in Bild 2 und 3 rechts neben dem Raster.

Schritt 4

Nachdem wir nun unsere Sprites dem Computer mundgerecht gemacht haben, überlegen wir uns, in welchen Speicherbereich wir sie plazieren. Zwei Bedingungen sind dabei zu beachten:

1) Die Sprites müssen im gleichen 16 K-Bereich liegen wie der Bildschirm. (Im Normalfall also von Speicherstelle 0 bis Speicherstelle 16383.)

2) Die Startadresse der Sprite-Daten muß glatt durch 64 teilbar sein.



Bit-Nr.	7	6	5	4	3	2	1	0
mal	128	64	32	16	8	4	2	1

Bild 4. Bitnummern und ihre Werte. Eine kleine Rechenhilfe.

Außerdem sollte man natürlich sinnvolle Speicherbereiche auswählen, das heißt solche, die weder der Computer selber braucht (wie die ersten 1023 Byte), noch solche, die für das Basic-Programm benötigt werden. Allerdings gibt es erfreulicherweise Lücken in den ersten 1023 Speicherplätzen, die nicht oder nur recht selten vom Computer benötigt werden. Man kann dort tatsächlich die Daten von vier Sprites ablegen:

704 bis 767 1. Sprite
832 bis 895 2. Sprite
896 bis 959 3. Sprite
960 bis 1023 4. Sprite

Die letzten Bereiche gehören zum Kassettenpuffer. Falls Sie also während eines Sprite-Programmes planen, eine Datasetten-Operation vorzunehmen, müssen Sie dafür sorgen, daß Ihre Sprite-Daten später wieder in den Kassettenpuffer geschrieben werden.

Sollten Sie mehr als vier Sprites benötigen, dann bleibt Ihnen nichts anderes übrig, als dazu den Basic-Speicherraum zu verwenden. Ich lege die Daten in diesem Fall häufig an die obere Grenze, also knapp unter 16383. Dann muß ich diesen Speicherteil vor dem Überschreiben durch das Basic-Programm (und Variable) schützen.

Eine andere Möglichkeit ist es, den Start des Basic-Speichers höher zu legen. All dies soll uns an dieser Stelle aber nicht belasten. Wir gehen hier davon aus, daß wir mit vier Sprite-Mustern auskommen.

Wie in der obigen Aufstellung schon angedeutet, legen wir unsere Sprite-Daten für Sprite 1 (das einfarbige Sprite) nach Speicherstelle 704 und für Sprite 2 nach 832.

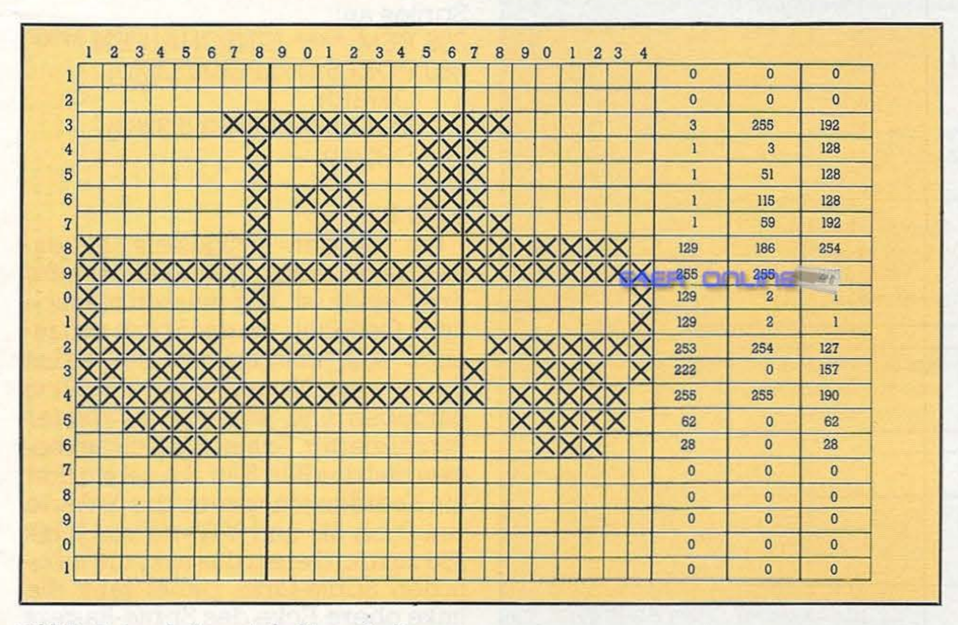


Bild 2. Entwurf eines einfarbigen Sprite

Zeile	Byte 1	Byte 2	Byte 3
1	128	8	0
2	160	40	0
3	168	168	0
4	40	168	0
5	42	170	0
6	10	234	160
7	43	250	128
8	43	250	0
9	170	232	0
0	170	168	0
1	165	168	0
2	129	90	0
3	129	86	0
4	0	80	128
5	0	80	0
6	0	16	20
7	0	16	85
8	0	17	84
9	0	21	0
0	0	16	0
1	0	16	0

Bild 3. So kann man einen Multicolor-Sprite aufbauen

Nun geht's ans Programmieren:

Schritt 5

Wir lesen die gewonnenen Sprite-Daten in den Computerspeicher ein mittels zweier simpler Schleifen:

```
100 FOR I=704 TO 766:READD
    :POKE I, D:NEXT I
110 FOR I=832 TO 894:READD
    :POKE I, D:NEXT I
115 REM ****** DATAS VON
    SPRITE 1 ******
120 DATA 0,0,0,0,0,0,3,255,192,1,3,128
130 DATA 1,51,128,1,115,128,1,59,192
140 DATA 129,186,254,255,255,255,
    129,2,1
150 DATA 129,2,1,253,254,127,222,0,
    157
160 DATA 255,255,190,62,0,62,28,0,28
170 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
175 REM ****** DATAS VON
    SPRITE 2 ******
180 DATA 128,8,0,160,40,0,168,168,0
190 DATA 40,168,0,42,170,0,10,234,160
200 DATA 43,250,128,43,250,0,170,
    232,0
210 DATA 170,168,0,165,168,0,129,90,0
220 DATA 129,86,0,0,80,128,0,80,0
230 DATA 0,16,20,0,16,85,0,17,84,0,
    21,0
240 DATA 0,16,0,0,16,0
```

Damit hätten wir den schlimmsten Teil der ganzen Arbeit hinter uns!

Schritt 6

Nun müssen wir unserem Computer mitteilen, an welchen Stellen im Speicher wir die Daten verstaut haben. Dazu gibt es ab Speicherstelle 2040 die sogenannten Sprite-Zeiger. Hier kommt die Startadresse der Daten hinein und zwar eine Kennzahl, die sich so ergibt:

Datenstartadresse/64 = Kennzahl

Für das erste Sprite folgt 704/64 = 11 und für das zweite Sprite 832/64 = 13. Gleichzeitig legt man auf diese Weise eine Sprite-Nummer fest, die von 0 (Speicherstelle 2040) bis 7 (Speicherstelle 2047) gehen kann. Unser Sprite 1 machen wir nun zum Sprite 0 und schreiben die Kennzahl 11 in Speicherstelle 2040. Analog dazu wird Sprite 2 zum Sprite 1 und wir schreiben 13 in Byte 2041:

```
245 REM ****
    SPRITE-ZEIGER ****
250 POKE2040,11:POKE2041,13
```

Immer dann, wenn im folgenden von einer Spritenummer N die Rede ist, meinen wir damit die durch die Spritezeiger festgelegte (also 0 für das einfarbige, 1 für das Multicolor-sprite).

Schritt 7

Ob wir diesen Schritt jetzt gehen oder später, ist eigentlich gleichgültig. Es dreht sich um das Einschalten

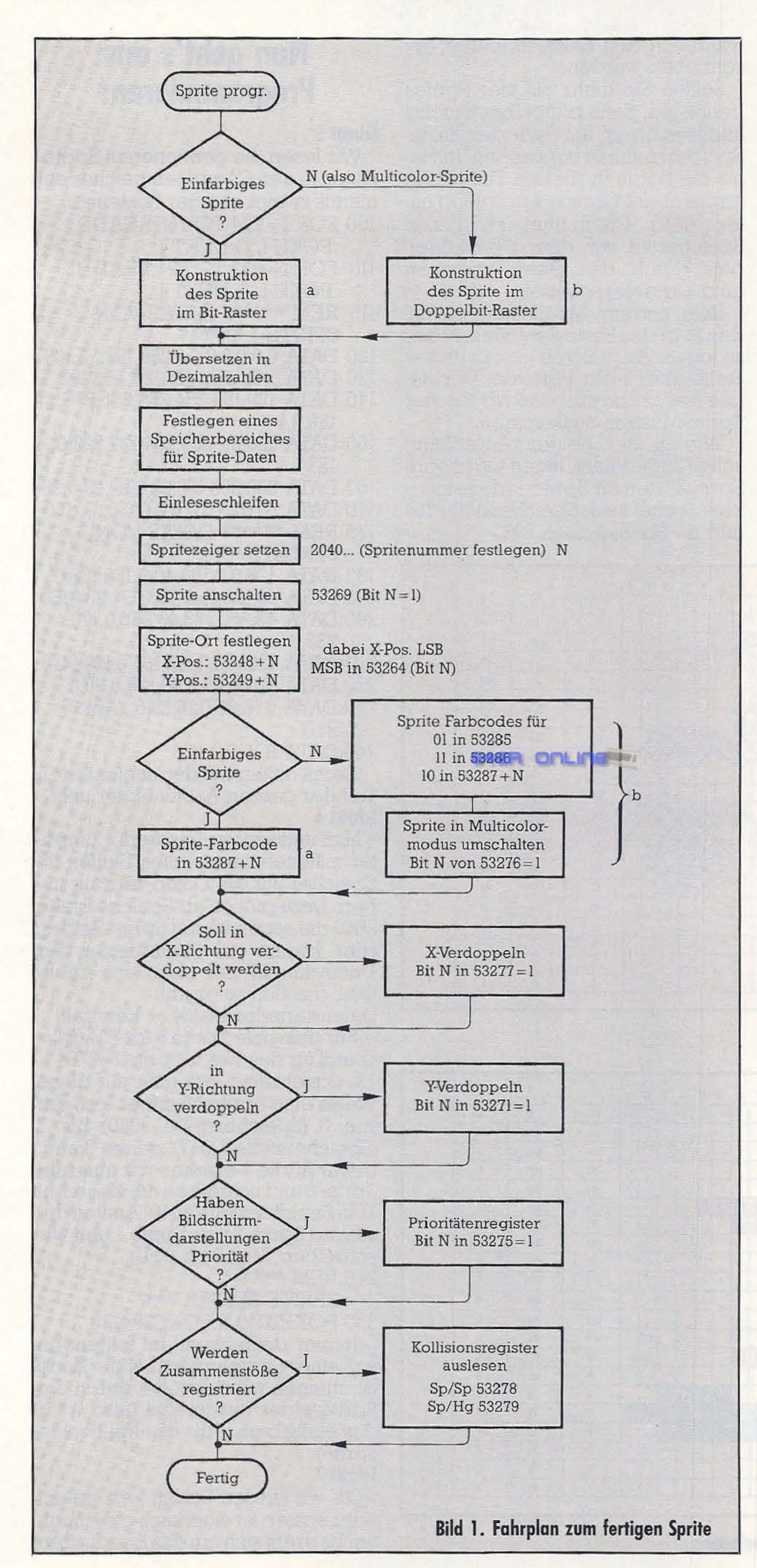


Bild 1. Fahrplan zum fertigen Sprite

der Sprites. Dazu bedienen wir uns des Registers in Speicherstelle 53269. Dies ist das erste einer Reihe von Registern, die bei Sprites eine Rolle spielen und nach einem Schema funktionieren: Von den 8 Bits des Registers gehört zu jedem Bit ein Sprite. Also zu Sprite 0 das Bit 0, zu Sprite 1 das Bit 1 und so weiter, allgemein zu Sprite N das Bit N. Jedes Bit mit dem Inhalt 1 bewirkt, daß das dazugehörige Sprite angeschaltet ist, ein Bit-Inhalt 0 sorgt für das Ausschalten des Sprite

Ohne nun auf mathematische Einzelheiten einzugehen, genügt es zu wissen, daß man einzelne Sprites einschaltet mittels:

```
POKE 53269,PEEK(53269)OR(2↑N)
```

und abschaltet mit:

```
POKE 53269,PEEK(53269)AND(255-2↑N)
```

Wir schalten nun unsere beiden Sprites an:

```
255 REM **** EINSCHALTEN ****
260 POKE53269,PEEK(53269)
    OR(2↑0):
    POKE53269,PEEK(53269)
    OR(2↑1)
```

Schritt 8

Sie werden festgestellt haben, daß kein Sprite auf dem Bildschirm erschienen ist. Wir müssen nämlich noch festlegen, wo das Sprite auftauchen soll, müssen also für jedes Sprite eine X- und eine Y-Position eingeben. Oje, jetzt wird es wieder komplizierter. Sehen Sie sich am besten mal das Bild 5 an. Sie sehen dort ein Koordinatensystem, das X-Werte von 0 bis 511 und Y-Werte von 0 bis 255 zuläßt. Genau das sind die möglichen Sprite-Orte. Dabei zählt die linke obere Ecke des Sprite-Rasters als Anhaltspunkt. Umrahmt ist der Teil der Fläche, der den sichtbaren Bildschirm darstellt. Man kann also Sprites so plazieren, daß sie unsichtbar bleiben, weil sie außerhalb des Bildschirms liegen. Auf dem Bild sind zwei Sprites gezeigt, von denen eins voll sichtbar (mit den Koordinaten 128, 120) und das andere unsichtbar ist (Koordinaten 360, 190). Auch teilweise sichtbare Sprites sind möglich.

Die X-Position eines Sprites wird ebenso wie die Y-Position in jeweils eine Speicherstelle gePOKEt. Diese findet man ab 53248, wobei 53248 die X-Position von Sprite 0, 53249 dessen Y-Position enthält, 53250 nimmt dann die X-Koordinate von Sprite 1, 53251 die Y-Koordinate auf und so weiter. Allgemein POKEt man die X-Koordinate für Sprite N in 53248 + 2*N und die Y-Koordinate in 53249 + 2*N.

Für die Y-Positionen ist das ohne Probleme. Bei der X-Koordinate tauchen Schwierigkeiten auf: Sobald eine X-Position größer als 255 benötigt wird, kann man sie nicht mehr einPOKEn. In diesem Fall verfährt man so. Man bildet die Differenz X-256 = X1 und POKEt X1 ein. Außerdem muß man nun in einem weiteren Register in 53264 (das wieder für jedes Sprite ein Bit bereithält) das dazugehörige Bit auf 1 setzen.

Für diese Eigenheiten gibt es natürlich eine Menge sehr eleganter Programmier-Möglichkeiten, die alle das Manko haben, reichlich unverständlich zu sein. Begnügen wir uns also mit einfachen IF..THEN-Abfragen. Unser Programm lautet also weiter:

```
265 REM **** SPRITE-
    POSITIONEN ****
270 FOR N=0TO1: PRINT
    "SPRITE"N;:
    INPUT"X,Y=";X(N),Y(N)
280 POKE 53249+2*N,Y(N)
290 IFX(N)>255THENPOKE53248
    +2*N,X(N)-256
    :POKE53264,PEEK(53264)OR
    (2↑N):GOTO305
300 POKE53248+2*N,X(N):
    POKE53264,PEEK(53264)AND
    (255-2↑N)
305 IF K=1 THEN RETURN
310 NEXT N
```

Dabei verlassen wir uns darauf, daß für X und Y nur sinnvolle Werte eingegeben werden. Wenn das aus irgendeinem Grund fraglich ist, müssen noch weitere IF...THEN-Abfragen dazukommen, die X und Y überprüfen vor Zeile 280. Die Zeile 305 ist vorsorglich für eine spätere Option eingebaut.

Schritt 9

Hier geht's um die Farbe unserer Sprites. Wie Sie sich denken können, müssen Multicolor-Sprites anders behandelt werden als einfarbige. Doch beiden gemeinsam ist zunächst folgendes:

Für jedes Sprite existiert ein Farbregister, das bei 53287 anfängt. Unser Sprite 0 hat in dieser Speicherstelle seinen Farbcode stehen, das Sprite 1 (das Multicolor-Sprite) in 53288 und so weiter. Allgemein steht die Farbe von Sprite N in Farbregister 53287+N.

Multicolorsprites bewahren in dieser Speicherstelle die Farbe auf, die zur Ziffernkombination 10 gehört. Die Kombination 01 und 11 werden für alle Multicolor-Sprites gemeinsam festgelegt. Dabei gehört der in 53285 stehende Farbcode zur Kombination 01, der in 53286 zur Kombination 11.

Außerdem muß für die Multicolor-Sprites noch der Mehrfarben-Modus eingeschaltet werden. Dazu existiert wieder ein Register (53276), in dem für jede Spritenummer N ein Bit reserviert ist. Ist dieses Bit gleich 1, dann liegt das dazugehörige Sprite im Mehrfarben-Modus vor. Allgemein schaltet man für Sprite N also den Multicolor-Modus an durch:

POKE 53276,PEEK(53276)OR(2↑N)

Das Rücksetzen geschieht durch Löschen des Bit N:

POKE 53276,PEEK(53276)AND(255-2↑N)

Wir wählen für Sprite 0 die Farbe CYAN, für das Multicolor-Sprite die Zuordnungen:

Kombination 01 = Grün
11 = Gelb
10 = Rot

```
315 REM ****
    SPRITE-FARBEN ****
320 POKE53287,3:REM SPRITE 0
    =CYAN
330 POKE53285,5:REM SPRITE 1
    KENNUNG 01 =GRUEN
340 POKE53286,7:REM
    KENNUNG 11=GELB
350 POKE53288,2:REM
    KENNUNG 10=ROT
360 POKE53276,PEEK
    (53276)OR(2↑1):
    REM MULTICOLORMODUS
    EINGESCHALTET
```

Schritt 10

Wenn Sie bis hierher unser schrittweise entwickeltes Programm eingegeben und gestartet haben, sehen Sie nun auf Ihrem Bildschirm zwei nette Sprites. Im folgenden werden wir nun noch einige Besonderheiten der Sprite-Programmierung kennenlernen, die wir aber im Text nur noch mit den allgemeinen Befehlen bearbeiten. Ein beigefügtes Beispielprogramm, das bis hierher (außer den Zeilen bis 100) mit unserem bisher entwickelten identisch ist, enthält die konkreten Befehle. Um das weitere Vorgehen deutlich zu machen, sind dort zunächst in den Zeilen 370 bis 400 drei weitere Sprites (mit denselben Daten) eingerichtet worden.

Schritt 10 stellt uns vor die Frage, ob wir unser Sprite in X-Richtung doppelt so groß darstellen wollen. Zu diesem Zweck gibt es dann wieder ein Register (jedes Bit ein Sprite), das Register 53277. Die X-Vergrößerung ist eingeschaltet, wenn für Sprite N das Bit N auf 1 gesetzt wurde mittels:

POKE53277,PEEK(53277)OR(2↑N)

Das Zurückschalten findet dann wieder statt mit ...AND(255-2↑N). Im Programm wurde Sprite 2 derart vergrößert.

Schritt 11

Dasselbe wie eben kann auch in der Y-Richtung geschehen. Dazu dient uns das Register 53271. Anschalten dieser Y-Verdoppelung also durch:

POKE53271,PEEK(53271)OR(2↑N)

und Abschalten wie oben schon gezeigt mittels ...AND(255-2↑N).

Im Programm ist Sprite 3 so vergrößert worden. Sprite 4 ist — auch das funktioniert — sowohl in X- als auch in Y-Richtung gedehnt worden.

Schritt 12

Hier geht's um die Vorfahrt. Wenn sich zwei Sprites überdecken, bleibt immer dasjenige »vorne«, das die niedrigere Nummer besitzt. Also verdeckt Sprite 0 alle anderen. Man hat darauf — außer durch geeignete Auswahl der Spritenummer ganz zu Beginn — keine Einflußmöglichkeit.

Anders verhält sich das bei der Vorfahrt von Sprites und Bildschirmzeichen. Die kann mittels Register 53275 geregelt werden. Wieder gehört zu jedem Sprite ein Bit. Ist dieses gleich 0, erscheint das Sprite vor den Bildschirmzeichen, ist es gleich 1, versteckt es sich dahinter. Im Beispielprogramm wurde in den Zeilen 450 und 460 geregelt, daß sich alle Autos (Sprites 0,2 und 4) hinter den Bildschirmzeichen aufhalten.

Schritt 13

Unser Computer paßt auch auf Zusammenstöße auf. Zwei Register (wieder Bit N für Sprite N) sind zuständig:

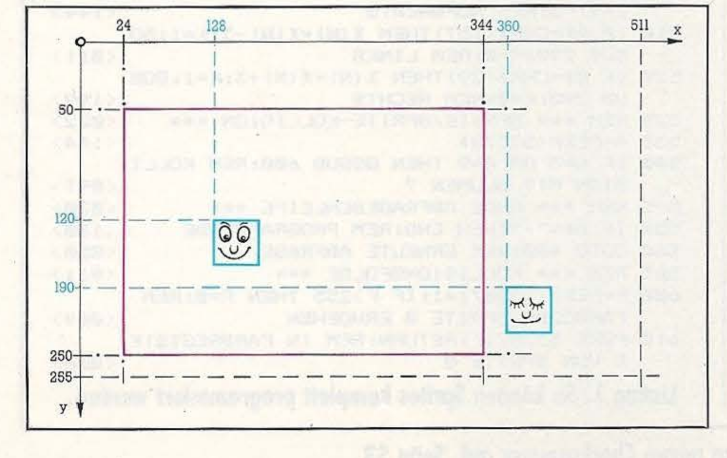


Bild 5. In diesem Feld können sich Sprites aufhalten. Rot umrandet, der sichtbare Bildschirm.

Kollisionen Sprite mit Sprite (53278), Kollisionen Sprite mit Bildschirmzeichen (53279). Für jedes Sprite, das in eine Kollision verwickelt wurde, wird das entsprechende Bit auf 1 gesetzt. Wenn also Sprite 0 mit Bildschirmzeichen zusammenstößt, findet man in Register 53279 den Wert 1. Die Berechnung des zu erwartenden Wertes kann mit dem Bild 4 geschehen, wobei für mehrere Sprites die Zahlen zu addieren sind. Im Beispielprogramm wird in den Zeilen 530 und 540 das Sprite/Sprite-Kollisionsregister 53278 auf Zusammenstöße von Sprite 0 mit Blumen (Sprite 1 und Sprite 3) abgefragt. Bei einer Kollision werden im Register 53278 die Werte 3 oder 9 erzeugt.

Das Auslesen dieser beiden Register ist ziemlich radikal: Nach dem PEEK-Kommando sind sie gelöscht. Deshalb sollte man die ausgelesenen Werte in einer Variablen speichern (Zeile 530:A). Außerdem kann man die Register durch PEEKen vor der ersten Kollisionsabfrage leeren, was wir in Zeile 460 machen.

Unser Beispielprogramm (Listing 1) ist so aufgebaut, daß wir mit den Cursortasten das Sprite 0 steuern können. Jedesmal, wenn dabei eine Blume überfahren wird, ändert sich die Autofarbe. Um das Programm kurz und übersichtlich zu gestalten, finden Sie keine Sicherung gegen das Heraussteuern aus dem zulässigen Sprite-Koordinaten-Bereich. Durch Eingabe von ← kann das Programm beendet werden.

Damit wissen Sie alles, was Sie zur Programmierung von Sprites brauchen. Probieren Sie mal ein wenig durch Veränderungen am Beispielprogramm herum. Falls Sie nun Blut geleckt haben, können Sie Ihr Wissen noch vertiefen durch die nachfolgend genannte Literatur:

— »Reise durch das Wunderland der Grafik«, 64'er, Ausgabe 8 (1984) Seite 142 ff. (Erscheint in diesen Tagen auch als Buch im Verlag Markt & Technik).

— H. Kunz hat etwas über das schnelle Bewegen von Sprites geschrieben im 64'er, Ausgabe 4 (1984) Seite 70 ff.

— Eine nette Anwendung stammt von H. Grigat in Happy-Computer, Ausgabe 11 (1983), Seite 99 ff.

— Einen Überblick geben Schneider und Ebert in den Bänden 1 und 3 des Commodore 64-Buches, erschienen im Verlag Markt & Technik (1984).

— Ebenfalls zu empfehlen ist S. Krutes »Grafik und Musik auf dem Commodore 64«, welches ebenfalls bei Markt & Technik erschien.

— Für alle, die es ganz genau wissen wollen (bis in die weiten Ebenen der Elektronik), empfehle ich schließlich noch den Aufsatz von P. Dornier »Sprites ohne Esoterik«, 64'er-Ausgabe 11 (1984), Seite 74 ff.

(Heimo Ponnath/ah)

```
1 REM *********************************    <132>
2 REM *                               *    <051>
3 REM *  BEISPIELPROGRAMM ZU SPRITES  *    <014>
4 REM *                               *    <053>
5 REM * HEIMO PONNATH   HAMBURG  1985 *    <130>
6 REM *                               *    <055>
7 REM *********************************    <138>
8 REM                                      <070>
45 REM **** BILDSCHIRMFARBEN ****          <156>
50 POKE 53280,0:POKE 53281,0:POKE 646,14   <072>
95 REM **** EINLESEN SPRITE-DATEN ****     <152>
100 FOR I=704 TO 766:READ D:POKE I,D:NEXT
    I                                      <135>
110 FOR I=832 TO 894:READ D:POKE I,D:NEXT
    I                                      <111>
115 REM ****** DATAS VON SPRITE 1 ******   <199>
120 DATA 0,0,0,0,0,0,3,255,192,1,3,128     <222>
130 DATA 1,51,128,1,115,128,1,59,192       <058>
140 DATA 129,186,254,255,255,255,129,2,1   <195>
150 DATA 129,2,1,253,254,127,222,0,157     <225>
160 DATA 255,255,190,62,0,62,28,0,28       <070>
170 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0     <028>
175 REM ****** DATAS VON SPRITE 2 ******   <132>
180 DATA 128,8,0,160,40,0,168,168,0        <118>
190 DATA 40,168,0,42,170,0,10,234,160      <185>
200 DATA 43,250,128,43,250,0,170,232,0     <164>
210 DATA 170,168,0,165,168,0,129,90,0      <174>
220 DATA 129,86,0,0,80,128,0,80,0          <143>
230 DATA 0,16,20,0,16,85,0,17,84,0,21,0    <087>
240 DATA 0,16,0,0,16,0                     <175>
245 REM **** SPRITE-ZEIGER ****            <142>
250 POKE 2040,11:POKE 2041,13              <243>
255 REM **** EINSCHALTEN ****              <151>
260 POKE 53269,PEEK(53269)OR(2↑0):POKE 532
    69,PEEK(53269)OR(2↑1)                  <045>
265 REM **** SPRITE-POSITIONEN ****        <079>
270 FOR N=0 TO 1:PRINT"SPRITE"N;:INPUT"X,Y
    =";X(N),Y(N)                           <120>
280 POKE 53249+2*N,Y(N)                    <188>
290 IF X(N)>255 THEN POKE 53248+2*N,X(N)-2
    56:POKE 53264,PEEK(53264)OR(2↑N):GOTO
    305                                    <102>
300 POKE 53248+2*N,X(N):POKE 53264,PEEK(53
    264)AND(255--2↑N)                      <136>
305 IF K=1 THEN RETURN                     <060>
310 NEXT N                                 <180>
315 REM **** SPRITE-FARBEN ****            <174>
320 POKE 53287,3:REM SPRITE 0 =CYAN        <053>
330 POKE 53285,5:REM SPRITE 1 KENNUNG 01=G
    RUEN                                   <058>
340 POKE 53286,7:REM KENNUNG 11=GELB       <167>
350 POKE 53288,2:REM KENNUNG 10=ROT        <228>
360 POKE 53276,PEEK(53276)OR(2↑1):REM MULT
    ICOLORMODUS EINGESCHALTET              <208>
365 REM *** SPRITE-GROESSEN ***            <095>
367 REM +++++ 3 WEITERE SPRITES +++++      <067>
370 POKE 2042,11:POKE 2043,13:POKE 2044,11
    :REM SPRITEZEIGER AUF VORHANDENE DATEN <149>
380 POKE 53269,PEEK(53269)OR(2↑2):POKE 532
    69,PEEK(53269)OR(2↑3)                  <245>
385 POKE 53269,PEEK(53269)OR(2↑4):REM ANSC
    HALTEN                                 <008>
390 POKE 53248+2*2,100:POKE 53249+2*2.100:
    POKE 53248+2*3,80:POKE 53249+2*3,200   <018>
395 POKE 53248+2*4,150:POKE 53249+2*4,150:
    POKE 53291,4                           <224>
400 POKE 53289,1:POKE 53290,6:POKE 53276,P
    EEK(53276)OR(2↑3):REM POSITIONEN+FARBE
    N                                      <170>
405 REM +++++ DIESE VERGROESSERN +++++     <236>
410 POKE 53277,PEEK(53277)OR(2↑2):REM SPRI
    TE 2 IN X-RICHTUNG VERDOPPELN          <077>
420 POKE 53271,PEEK(53271)OR(2↑3):REM SPRI
    TE 3 IN Y-RICHTUNG VERDOPPELN          <245>
430 POKE 53271,PEEK(53271)OR(2↑4):POKE 532
    77,PEEK(53277)OR(2↑4):REM SPRITE 4 X+Y <241>
435 REM **** VORFAHRT-REGELUNG *****       <079>
440 POKE 53275,PEEK(53275)OR(2↑0):POKE 532
    75,PEEK(53275)OR(2↑2)                  <131>
450 POKE 53275,PEEK(53275)OR(2↑4):REM ALLE
     AUTOS HINTER BILDSCHIRMZEICHEN        <245>
455 REM *** KOLLISIONEN VORBEREITEN ***    <193>
460 A=PEEK(53278):A=PEEK(53279)            <243>
465 REM *** STEUERN VON SPRITE 0 ***       <151>
470 PRINT CHR$(147):N=0                    <058>
480 GET A$:IF A$=""THEN 480                <197>
490 IF A$=CHR$(17)THEN POKE 53249,PEEK(532
    49)+3:REM ABWAERTS                     <252>
500 IF A$=CHR$(145)THEN POKE 53249,PEEK(53
    249)-3:REM AUFWAERTS                   <144>
510 IF A$=CHR$(157)THEN X(N)=X(N)-3:K=1:GO
    SUB 290:K=0:REM LINKS                  <011>
520 IF A$=CHR$(29)THEN X(N)=X(N)+3:K=1:GOS
    UB 290:K=0:REM RECHTS                  <199>
525 REM *** SPRITE/SPRITE-KOLLISION ***    <022>
530 A=PEEK(53278)                          <144>
540 IF A=3 OR A=9 THEN GOSUB 600:REM KOLLI
    SION MIT BLUMEN ?                      <091>
545 REM *** ENDE ABFRAGESCHLEIFE ***       <070>
550 IF A$="←"THEN END:REM PROGRAMMENDE     <130>
560 GOTO 480:REM ERNEUTE ABFRAGE           <050>
565 REM *** KOLLISIONSFOLGE ***            <011>
600 F=PEEK(53287)+1:IF F>255 THEN F=0:REM
    FARBCODE SPRITE 0 ERHOEHEN             <089>
610 POKE 53287,F:RETURN:REM IN FARBREGISTE
    R VON SPRITE 0                         <086>
```

Listing 1. So können Sprites komplett programmiert werden.

Beachten Sie die Eingabehinweise zum neuen Checksummer auf Seite 53.

Das Drum-Kit von Compware, bestehend aus Hard- und Software, ist eine kleine Sensation. Die Schlaginstrumente, die damit simuliert werden können, klingen fast wie die eines echten Schlagzeugs. Auch wir in der Redaktion konnten uns nicht der Faszination des Klanges entziehen.

Das Funktionsprinzip des Drum-Kits ist eigentlich ganz einfach. Die Tonsignale eines Schlagzeugs wurden in einem Tonstudio aufgenommen und digitalisiert. Die digitalen Werte sind dann mit einem Steuerungsprogramm für die Tastaturabfrage zusammen auf Diskette gespeichert worden.

Hat man das Programm geladen, können die einzelnen Schlaginstrumente über das Tastenfeld des C 64 gespielt werden. Wie schon erwähnt, ist zum Programm noch etwas Hardware nötig: ein D-A-Wandler auf einer kleinen Platine, der am User-Port angeschlossen wird. Durch den Wandler werden die gespeicherten digitalen Frequenzwerte eines Schlagzeugs wieder in analoge hörbare Signale umgeformt. Für die Wiedergabe stehen zwei verschiedene Ausgangsbuchsen (eine 5polige DIN- und eine Klinkenbuchse), an die ein Verstärker angeschlossen werden muß, zur Verfügung.

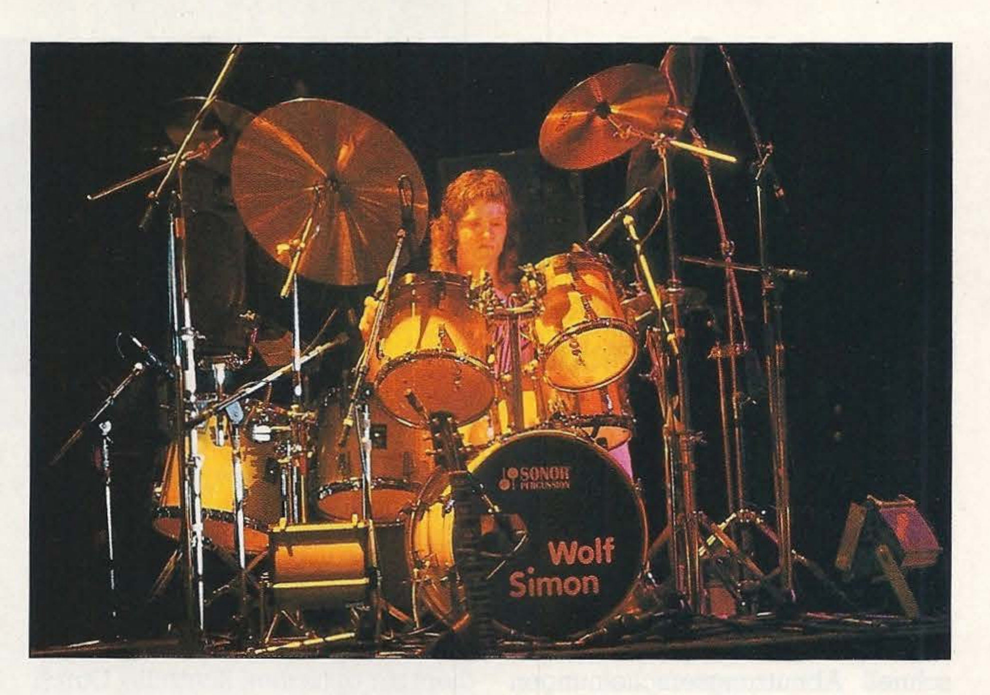
Das Steuerungsprogramm erlaubt drei Arten der Bedienung:

1. Man kann über die beiden oberen Tastenreihen spielen. Zwei übereinanderliegende Tasten wurden dem gleichen Schlaginstrument zugeordnet, damit auch schnelle Trommelwirbel (jeweils abwechselnd obere und untere Taste drücken) möglich sind.

An Instrumenten stehen zwei Baßtrommeln, zwei Snare Drums, vier Tom-Toms, zwei Hi-Hats, ein Rim-Shot, zwei Becken und eine Kuhglocke zur Verfügung. Allerdings können nicht alle über die Tastatur gespielt werden.

Programmierter Rhythmus

2. Auch mit dem Joystick kann gespielt werden, wobei pro Port fünf Instrumente verfügbar sind. Mit dieser Art der Steuerung sind jedoch keine so guten Spielergebnisse wie auf der Tastatur zu erzielen, da die Instrumente durch eine Repeatfunktion sehr schnell angeschlagen werden. Das klingt dann wie ein Maschinengewehrfeuer oder eine ratternde Nähmaschine.



Trommelwirbel

Eine kleine Platine und ein Programm auf Diskette machen aus dem C 64 einen Schlagzeug-Computer, der einen Klangvergleich mit einem »normalen« Schlagzeug fast nicht zu scheuen braucht.

3. Ein Schlagrhythmus kann auch in Basic programmiert werden. Nach dem Programmstart mit RUN trommelt der C 64 los. Dazu wurde der Basic-Befehlssatz um eine »Schlagzeugsprache« erweitert. So spricht man zum Beispiel die große Baßtrommel über den Befehl »$B1« an. Auch eine Pausefunktion ist implementiert; sie wird mit »$P« aufgerufen. Alle neuen Befehle können zusammen mit dem normalen Basic verwendet werden. So kann ein Rhythmus über eine FOR...NEXT-Schleife wiederholt werden. Eine Taktprogrammierung könnte vielleicht so aussehen:

```
10 FOR T=1 TO 3: $B1: $P: $S1: $S1:
$P: NEXT
```

Die Trommel-Geschwindigkeit ist durch zwei POKE-Befehle (752 und 753) einstellbar. Auf der mitgelieferten Diskette befindet sich ein Demoprogramm, das einige Rhythmusbeispiele enthält.

Als Erweiterung für das Digi-Drum-Kit gibt es eine Platine für den Joystickport, an die sich Pads anschließen lassen. Pads sind elektronische Schlagsensoren, auf denen man wie mit einem Schlagzeug spielen kann. Ein Pad ist prinzipiell ein Mikrofon in einem meist fünf- und sechseckigen Gehäuse. Wird auf ein Pad geschlagen, so entsteht durch das Mikrofon ein elektrischer Impuls, der von einem Synthesizer weiterverarbeitet werden kann. Auf diese Weise können synthetische Schlaggeräusche erzeugt werden. Eine Diskette mit Simons-Sounds, synthetischen Schlaggeräuschen ist auch erhältlich.

Der C 64 — ein professionelles Schlagzeug?

Das Drum-Kit von Compware ist zwar kein vollwertiger Ersatz für ein Schlagzeug, aufgrund des echten Klanges könnte es allerdings dennoch professionell genutzt werden. Zumal es auch einfach zu programmieren ist. Ob man sich als Normalbenutzer — ohne musikalische Ambitionen — diese Erweiterung, die zumal nicht ganz billig ist, anschaffen soll, muß man letztendlich selbst entscheiden.

(Christoph Sauer/hm)

Digital Drums, Helmut Adler Computer-Technologie, Schlägel & Eisen-Str. 9, 4352 Herten-Langenbochum, Preis 179,— Mark

Dauerfeuer

Ist Ihr Zeigefinger bei manchen Spielen auch zu langsam? Wenn Sie einen Joystick ohne Dauerfeuer haben, können Sie diese Funktion leicht nachrüsten.

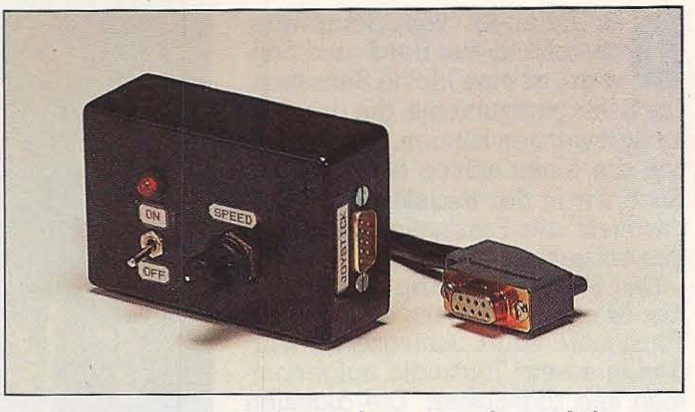


Bild 1. Die fertige Dauerfeuer-Schaltung schont Daumen und Zeigefinger

Leider ist es bei manchen Actionspielen notwendig, wie ein Wilder auf seinem Feuerknopf herumzutrommeln, um ein einigermaßen hohes Punktekonto zu erhalten. Dies tut weder dem Joystick, der während einer solchen Prozedur schnell Abnutzungserscheinungen zeigt, noch der Hand des Spielers gut. Die Industrie, die ihre Chance schnell erkannte, brachte deshalb Joysticks mit sogenannten Repeat-Schaltungen (auch Autofire genannt) auf den Markt. Dadurch ist es nicht mehr nötig, den Feuerknopf in schneller Folge zu drücken. Das wird von der eingebauten Elektronik übernommen. Spiele wie Fort-Apocalypse oder Scramble werden dann zum Genuß. Aber was macht man, wenn man sich gerade erst einen teuren Joystick ohne Repeat-Schaltung gekauft hat oder das nötige Kleingeld fehlt? Nun, genau dafür wurde diese Repeat-Schaltung gebastelt, mit der Joysticks nachgerüstet werden können. Da der Aufbau keine besonderen Ansprüche stellt, sollte er auch vom Ungeübten zu bewältigen sein. Man muß nur mit einem Lötkolben umgehen können. Besonders komfortabel wird die Schaltung durch die Möglichkeit über ein Potentiometer die Frequenz, die Geschwindigkeit mit der die Feuersignale erzeugt werden, einzustellen. Eine Leuchtdiode dient zur optischen Kontrolle. Durch einen Schalter läßt sich die Schaltung ein- oder ausschalten. Im letzten Fall verhält sich Ihr Joystick wieder völlig normal. Um einen Eingriff in den Computer oder den Joystick zu vermeiden, hat sich die Methode, das »Autofire« einfach zwischen beide zu schalten, bewährt. Wer Lust hat, kann sich ein formschönes Gehäuse besorgen, um die Schaltung reizvoll zu verstecken (Bild 1).

Das Herzstück ist der Timer NE555, der Pin 6 der Joystickbuchse am Computer periodisch auf Masse legt. Die Geschwindigkeit (Frequenz) wird über ein RC-Glied, bestehend aus dem 0,1-µF-Kondensator und dem 25-kΩ-Poti, eingestellt. Mit dem Feuerknopf wird der Timer an Masse angeschlossen und das Feuersignal periodisch auf Masse gelegt.

Wie schon erwähnt, ist der Aufbau einfach und unkritisch. Fehler in der Verdrahtung sind kaum zu machen. Trotzdem sollte dem IC ein Sockel spendiert werden. Die Bauteile wurden so gewählt, daß es keinerlei Bezugsschwierigkeiten geben dürfte. Am besten, man baut die Schaltung auf einer Lochrasterplatine auf. Das ist einfach und geht bei dieser kleinen Schaltung auch schneller als das Ätzen und Bohren einer eigenen Platine. Den Schaltplan der Dauerfeuer-Schaltung finden Sie in Bild 2, die Stückliste in Tabelle 1. Die Anschlußbelegung des NE555 zeigt Bild 3. Ursprünglich wurde die Schaltung für den C 64 entwickelt. Wie ein Test erwies, läßt sie sich aber auch problemlos am VC 20 betreiben.

Achtung: Im eingeschalteten Zustand des Computers sollten Sie die Repeat-Schaltung, wie alle anderen Peripheriegeräte auch, niemals in den Computer ein- oder ausstecken. Nur so gehen Sie sicher, ihn durch eventuelle statische Ladungen nicht zu beschädigen.

(Jörg Huth/hm)

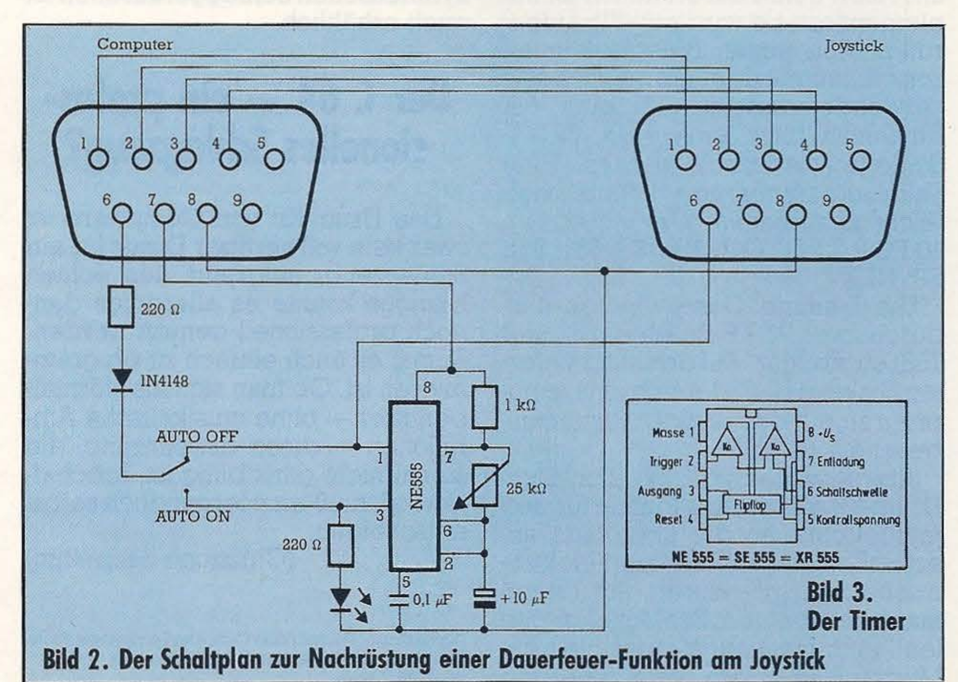
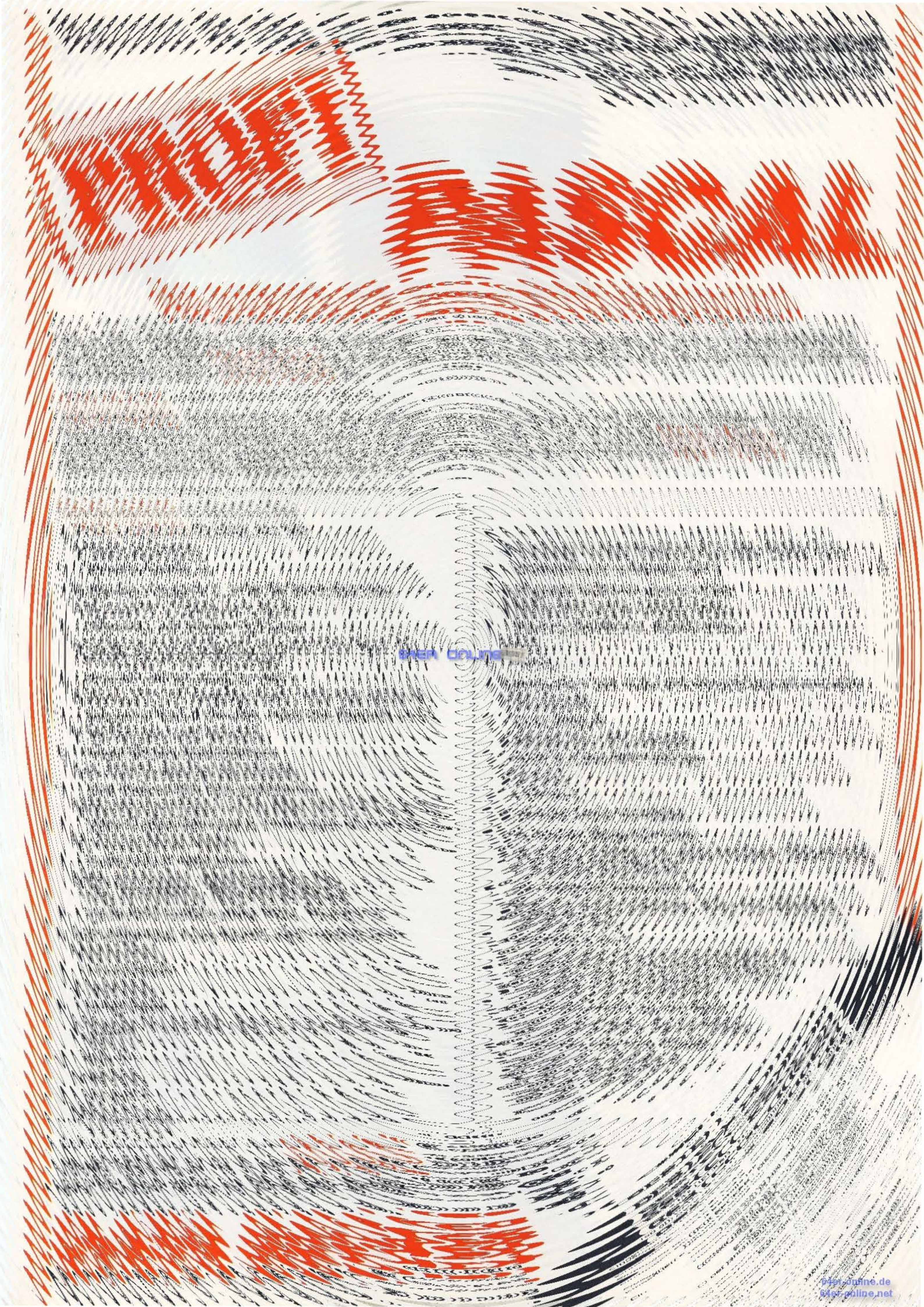


Bild 3. Der Timer

Bild 2. Der Schaltplan zur Nachrüstung einer Dauerfeuer-Funktion am Joystick

Stückliste:
1 Stecker + 1 Buchse für Joystick-Controlport
1 IC NE555 + 1 Sockel (8polig)
1 Diode 1N4148
Widerstände:
2 x 220 Ohm
1 x 1 kOhm
1 x 25 kOhm Poti
Kondensatoren:
1 x 10 µF
1 x 100 nF
1 LED (5 mm)
1 Schalter (1 x Um)
1 Stück Lochrasterplatine (Lötpunkte; 2,54 mm Rasterabstand)

Tabelle 1. Stückliste. Die Bauteilkosten betragen etwa 15 Mark



Trends und Flops

Ankündigungen für Rollenspielfans: Im Anmarsch sind »Ultima IV — Quest of the Avatar« und »Möbius I — The Orb of Celestial Harmony« vom Autor des Klassikers »Ultima III«. Spinnaker Software bietet in Kürze ein gewaltfreies Rollenspiel an, das in einer Welt spielt, in der man Gedanken lesen kann. Titel: »Below the Root«. »Xyphus«, ein im Ultima-Stil geschriebenes Programm, wird von Penguin Software jetzt auch für den C 64 angeboten. Auch Adventure-Fans werden voll auf ihre Kosten kommen: Synapse Software und Broderbund bieten gemeinsam eine Reihe von Adventures an, die unter dem Markenzeichen »Electronic Novels« kommen werden. Sie sollen laut US-Fachpresse sogar die Infocom-Adventures übertreffen: Der Grundwortschatz beträgt 1200 bis 1500 Worte, die Abenteuer spielen in sogenannten Echtzeit-Universen, die sich ständig ändern (Politik und Generationswechsel inbegriffen). Erhältlich sind schon »Mindwheel« und »Essex«, die Adventures »Brimstone«, »Breakers« und »Ronin« sind in Arbeit. Sobald diese Spiele in Deutschland lieferbar sind, werden wir darüber berichten.

Microprose Software will eine neue Serie von Strategiespielen anbieten. Die Werbung für diese Spiele spricht schon von der »nächsten Generation der Strategie-Simulationen«. Die ersten Titel aus dieser sogenannten »Command Series« sind »Crusade in Europe« und »Decision in the Desert«. Sobald wir mehr über die Besonderheiten der »nächsten Generation der Strategiespiele« wissen, werden wir uns damit beschäftigen.

Der Sportspiele-Trend flaut allmählich ab; mittlerweile haben eben doch einige Firmen eingesehen, daß die Olympiade schon vorbei ist. Nur noch wenige Sportspiele sind im Anmarsch: Epyx bringt »Summer Games II« und »Two on Two«, Gamestar kündigte »On Track Racing« an, und Cosmi will mit »Richard Petty's Talladega« eine (eher mißglückte) 3D-Stock-Car-Racing-Simulation an den Mann bringen.

Flops des Monats sind diesmal »Web Dimension«, ein furchtbar langweiliges Spiel von Activision, sowie »Megazone« von Data Media. Hierbei handelt es sich um einen sehr langsamen Fort Apocalypse-Verschnitt, bei dem einzig das Titelbild gelungen ist.

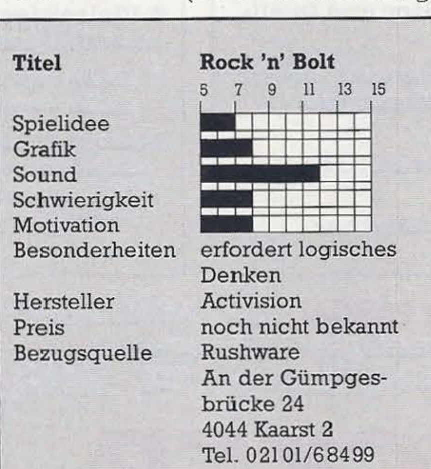
(M. Kohlen/B. Schneider/rg)



Rock 'n' Bolt

Wieder einmal liegt mit Rock'n' Bolt ein Activison-Spiel vor, das durch seine sehr gute Musik-Begleitung besticht. Rock 'n' Bolt kann man als Intelligenzspiel bezeichnen. Es geht darum, sich bewegende Stahlträger mit Bolzen zu sichern. Das ist schwerer, als es auf den ersten Blick aussieht. Man darf sich dabei nicht den Rückweg zum Ausgangspunkt verbauen, denn sonst verliert man wertvolle Zeit beim Ent- und wieder Verbolzen der einzelnen Träger. Man spielt, wie so oft, gegen die Uhr. Der Lohn der Arbeit wird in harten Dollars — leider nur auf dem Bildschirm — ausbezahlt. Hierbei geht die Anzahl der Bolzen und die übriggebliebene Zeit in die Bewertung ein. Bonus-Dollar und -Leben lassen sich durch Berühren farblich gekennzeichneter Bolzen erreichen.

Damit nicht genug: In den höheren Screens beginnt der Computer vorzuschreiben, wo die einzelnen Träger festgebolzt werden müssen. Dann erstrecken sich die zu bearbeitenden Flächen über mehrere Bildschirme. Nun aber zur Musik. Was dem Spieler im Rock'n'Roll-Stil aus dem Lautsprecher entgegenklingt, schlägt unserer Meinung nach sogar Ghostbusters. Die Grafik ist recht witzig, schnell und sauber. Ein Spiel, das wir jedem »Intelligenzbolzen« wärmstens empfehlen können. (Boris Schneider/rg)



Titel	Rock 'n' Bolt
	5 7 9 11 13 15
Spielidee	
Grafik	
Sound	
Schwierigkeit	
Motivation	
Besonderheiten	erfordert logisches Denken
Hersteller	Activision
Preis	noch nicht bekannt
Bezugsquelle	Rushware An der Gümpgesbrücke 24 4044 Kaarst 2 Tel. 02101/68499



Abenteuer-Paket 1

Der Trend zur preiswerten Software ist immer stärker zu spüren. Der Markt & Technik-Verlag folgt diesem Trend mit einer Serie von Abenteuer-Paketen.

Die Pakete enthalten je zwei Spiele, die sich über vier Diskettenseiten erstrecken. Für unsere Besprechung ziehen wir jeweils das beste Adventure heran.

Sagor der Eroberer

Dieses Abenteuerspiel zeichnet sich durch seine hervorragende Grafik aus. 27 ganzseitige HiRes-Bilder erfreuen das Auge, stellen aber die Geduld auf eine harte Probe. Über 30 Sekunden dauert das Nachladen eines Bildes. Ein Programm-Neustart nimmt fast fünf (!) Minuten in Anspruch (ohne Lesen der Beschreibung).

Dieses Abenteuerspiel verfügt über einen variablen Spielverlauf und einige besondere Gags, über die hier nichts verraten werden soll. Insgesamt erstreckt sich dieses Spiel über drei Diskettenseiten.

Alles in allem ist Sagor der Eroberer aber ein Abenteuerspiel, das eindeutig über dem Durchschnittsadventure liegt. Ein logischer Spielverlauf und einige besondere Gags im Zusammenhang mit der Grafik lassen den Spielspaß lange anhalten.

Noch auf der Diskette enthalten: Operation Neptun. (rg)



Titel	Sagor der Eroberer
	5 7 9 11 13 15
Spielidee	
Grafik	
Sound	
Schwierigkeit	
Motivation	
Besonderheiten	ganzseitige HiRes-Grafik, 2 Adventures in einem Paket
Hersteller	Markt&Technik Verlag
Preis	34,90 Mark
Bezugsquelle	Markt&Technik Verlag AG, Hans-Pinsel-Str. 2, 8013 Haar bei München

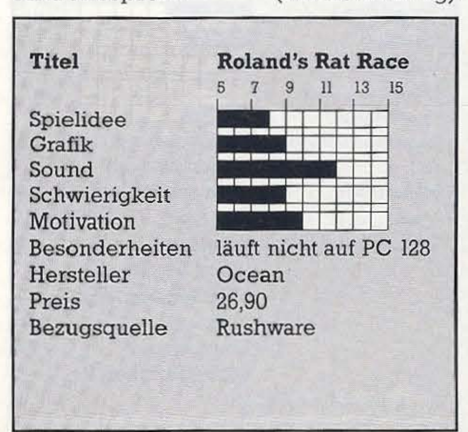


Rolands Rat Race

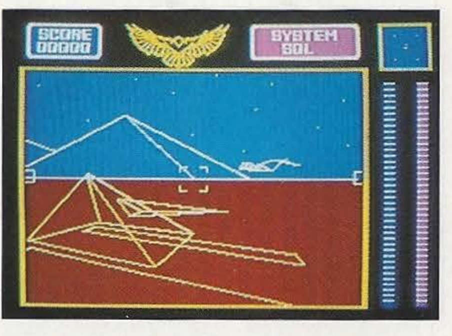
Roland Ratte, der den Briten jeden Morgen im Fernseh-Vormittagsprogramm begegnet, ist nun auch auf den C 64-Bildschirm gekrabbelt. Dummerweise hat er verschlafen, sein Auto ist kaputt, und er muß unbedingt um 9 Uhr im Fernsehstudio sein. Doch das ist nicht das einzige Problem. Die bösen Treter haben nämlich etwas gegen unseren lieben Roland.

Roland kennt sich bestens im Untergrund aus und muß den Weg nach oben finden. Vorher muß er allerdings die Teile der Ausgangstür aufsammeln, die im Londoner Untergrund verteilt sind. Roland darf natürlich während seiner Unterwelt-Odyssee nicht schlappmachen, denn er hat nur eine gewisse Menge an Energie. Trifft er auf Treter (oder besser gesagt sie auf ihn), verliert er Energie. Um das zu vermeiden, kann er allerdings seine Superkleber-Kanone benutzen. Die Treter bleiben am Leim hängen und Roland kommt vorbei, ohne einen Tritt in der Hintern zu bekommen. Seine Energie aufüllen kann Roland durch herumliegende Äpfel, Hamburger etc. Diese sind allerdings nur sehr spärlich im Untergrund verteilt.

Die Grafik des Spiels kann überzeugen, und die Musik ist wohl das beste, was seit langem in einem Spiel zu hören war. Die Spielidee ist zwar nichts Neues, macht aber doch eine gewisse Weile lang Spaß. Ein »Such-Sammel-Abliefer-Labyrinth-Aktionsspiel«. (M. Kohlen/rg)



Titel	Roland's Rat Race
Spielidee	
Grafik	
Sound	
Schwierigkeit	
Motivation	
Besonderheiten	läuft nicht auf PC 128
Hersteller	Ocean
Preis	26,90
Bezugsquelle	Rushware



Stellar 7

Unser Klasssiker

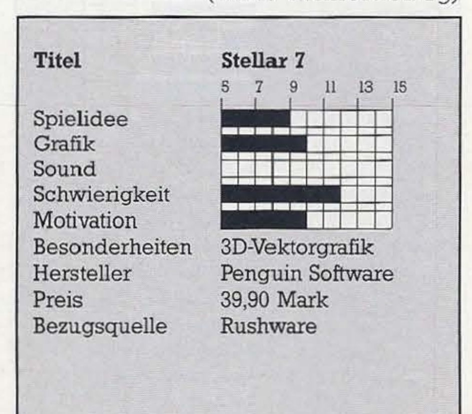
Auf dem Spiele-Markt gibt es einige »Dauerbrenner«, die sich auch noch nach zwei oder drei Jahren verkaufen. Beispiele sind »Loderunner«, »M.U.L.E.« oder »Summergames«. Grund genug für uns, monatlich ein etwas älteres Programm vorzustellen. Den Anfang machen wir mit dem 3-D-Action-Renner »Stellar 7«, der gerade in einer preiswerten Neuauflage erschienen ist.

Zur Handlung: Die arkturische Kriegsflotte will das Sonnensystem erobern. Nur Sie können das verhindern, indem Sie mit Ihrem Kampfgleiter das arkturische Flaggschiff zerstören. Der Weg dorthin führt auf die Planeten von sieben Sternsystemen, die nacheinander besucht werden müssen. Dabei ist die Vielfalt dieser Gegner verblüffend: Verschiedene Typen von Panzern, Fliegern und festen Geschützen tummeln sich auf den Planeten. Damit ist nicht nur gutes Reaktionsvermögen, sondern auch eine ordentliche Portion Strategie nötig, will man Arkturus lebend erreichen. Einstellbare Schwierigkeitsstufen verhelfen sowohl Anfängern wie Profis zu spannenden Stunden und Erfolgserlebnissen.

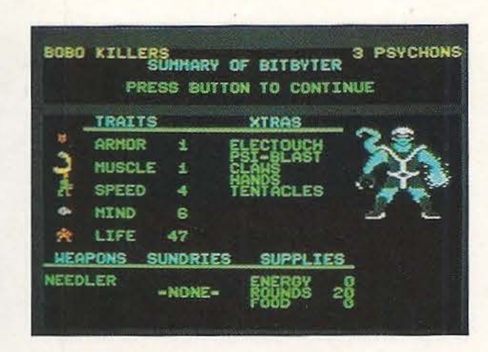
Die 3D-Vektor-Grafik ist wirklich beeindruckend, dafür ist der Sound recht spärlich.

Wer mal wieder seinen Feuerknopf und Kopf gebrauchen will, sollte Stellar 7 spielen.

(Boris Schneider/rg)



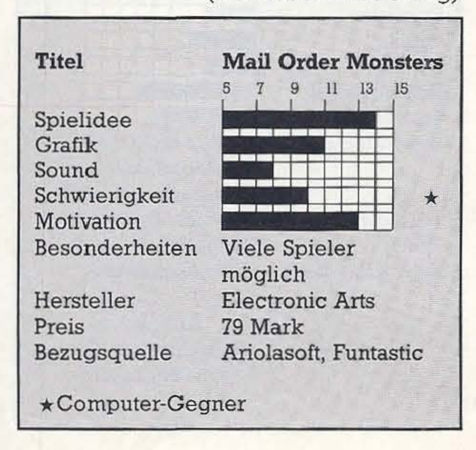
Titel	Stellar 7
Spielidee	
Grafik	
Sound	
Schwierigkeit	
Motivation	
Besonderheiten	3D-Vektorgrafik
Hersteller	Penguin Software
Preis	39,90 Mark
Bezugsquelle	Rushware



Mail Order Monsters

Mit Mail Order Monsters liegt wiederum ein Spiel vor, bei dem zwei Mitspieler ihre Aggressionen aneinander auslassen können. Jeder der beiden bekommt vom Computer 250 Psychons (Währungseinheit) in die Hand gedrückt, und darf sich damit ein Monster kaufen und ausrüsten. Dabei besteht die Auswahl zwischen zwölf verschiedenen Grundformen. Man darf die Eigenschaften eines Monsters, von körperlicher Stärke bis zur Intelligenz, verbessern, genügend Geld vorausgesetzt. Schließlich hat man noch die Auswahl zwischen 15 Waffen, 6 Defensivmitteln und 9 Dingen, die unter »Sonstiges« fallen. Hat jeder sein Monster fertiggestellt und ausgerüstet, darf gekämpft werden. Jeder Spieler steuert sein Monster per Joystick möglichst so, daß es gewinnt und eine Siegesprämie mit nach Hause nehmen kann, die dann natürlich für weitere Monster, Waffen etc. genutzt werden kann. Maximal können gleichzeitig zwei Spieler gegeneinander kämpfen, doch verwaltet der Computer beliebig viele Spielerkonten, so daß durchaus monatelange Turniere mit mehreren Dutzend Spielern möglich sind. Die Grafik ist größtenteils umwerfend gut gelungen, in den Zweikämpfen selbst allerdings etwas schlicht. Wer ein abwechslungsreiches Spiel für zwei oder mehr Spieler sucht, wird von Mail Order Monsters begeistert sein.

(Boris Schneider/rg)



Titel	Mail Order Monsters
Spielidee	
Grafik	
Sound	
Schwierigkeit	
Motivation	★
Besonderheiten	Viele Spieler möglich
Hersteller	Electronic Arts
Preis	79 Mark
Bezugsquelle	Ariolasoft, Funtastic

★Computer-Gegner



Australopiticus Robustus

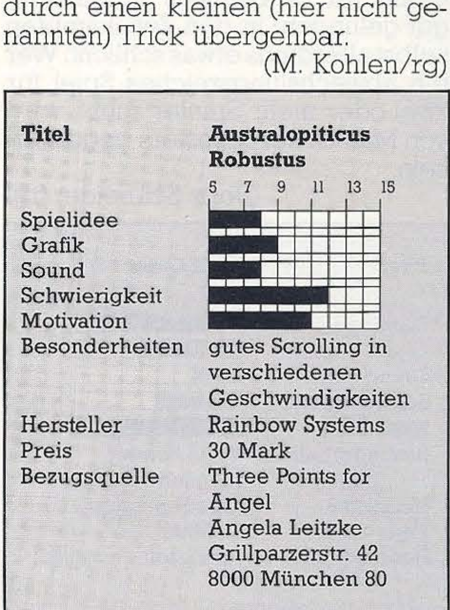
Australopiticus Robustus ist der Name einer Urzeitmenschen-Gattung. Ziel des Spiel ist es, drei Steinzeitmenschen, die Brüder Org, Fred und Gnom, durch ein gefährliches Labyrinth zu führen. Das Labyrinth besteht aus zehn verschiedenen Bildern, von denen jedes 2040 Pixels breit ist und in verschiedenen Geschwindigkeiten über den Bildschirm scrollt.

Das Problem besteht nun darin, die Levels zu verlassen. Dazu muß man sich aber erst die passenden Stücke der nötigen Leiter zusammensuchen. Der Haken an dieser Sache ist wiederum das Vorhandensein einiger blutrünstiger Dinosaurier, die alles tun, um zu ihrer Leibspeise (Mensch) zu kommen.

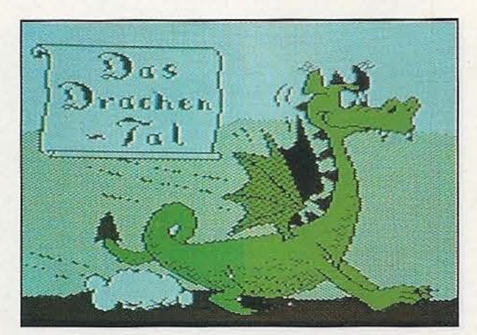
Der Sound des Spieles ist zwar gut gemacht, wirkt aber auf die Dauer eintönig. Ein großer Pluspunkt gebührt dem Spiel für die gelungene Grafik. Die Animation ist durchwegs stufenlos, die Farbkombinationen sind gut gewählt und die Knollennasen fallen auf.

Der Spielwert ist gut; man hat immer den Drang, nach einem beendeten Spiel neu zu beginnen, um es doch noch zu schaffen. Der ziemlich hohe Schwierigkeitsgrad ist dabei durch einen kleinen (hier nicht genannten) Trick übergehbar.

(M. Kohlen/rg)



Titel: Australopiticus Robustus
5 7 9 11 13 15
Spielidee
Grafik
Sound
Schwierigkeit
Motivation
Besonderheiten: gutes Scrolling in verschiedenen Geschwindigkeiten
Hersteller: Rainbow Systems
Preis: 30 Mark
Bezugsquelle: Three Points for Angel, Angela Leitzke, Grillparzerstr. 42, 8000 München 80



Abenteuer-Paket 2

Von den Abenteuer-Spielen des zweiten Abenteuer-Paketes ist das Drachental besonders bemerkenswert.

Das Drachental

Mit schön und witzig gemachter HiRes-Grafik wird der Abenteurer bei diesem Spiel überrascht. Einige kleine, zeichentrickähnliche Animationen und besondere Spielsituationen runden den durchweg positiven Eindruck vom Drachental ab.

Besonders zu erwähnen: die HiRes-Grafiken der zuletzt betretenen Räume bleiben im Speicher erhalten und müssen bei einem eventuellen Zurückgehen nicht erneut geladen werden.

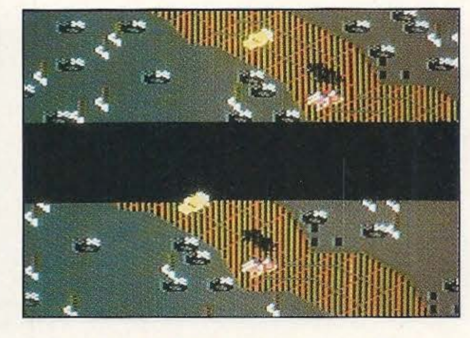
Die Aufgabe: Der Drache »Eusebius« verwüstet immer wieder das Zwergenland. Sie sollen den Drachen unschädlich machen; und damit ist nicht gemeint, daß Sie ihn töten sollen. Drachental ist ein recht friedliches Adventure. Auf dem Weg zur Lösung muß der Spieler oft zu ungewöhnlichen Maßnahmen greifen. Ein Tip: Achten Sie auf die Seife!

Das Drachental ist sicher eine Bereicherung des Adventure-Marktes und wird Abenteurern bestimmt viel Spaß machen.

Noch auf der Diskette enthalten: Flucht ins Paradies. (rg)



Titel: Das Drachental
5 7 9 11 13 15
Spielidee
Grafik
Sound
Schwierigkeit
Motivation
Besonderheiten: zeichentrickähnliche Animation, 2 Adventures in einem Paket
Hersteller: Markt&Technik Verlag
Preis: 34,90
Bezugsquelle: Markt&Technik Verlag AG, Happy Software, Hans-Pinsel-Str. 2, 8013 Haar bei München



Racing Destruction Set

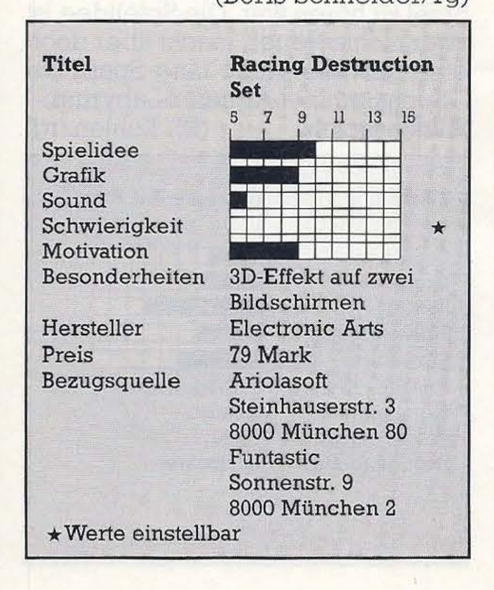
Das Racing Destruction Set ist das erste dreidimensionale Autorennen: Es geht nicht nur links- und rechtsrum, sondern auch rauf und runter. Gefahren werden kann zu zweit oder allein gegen den Computer auf zwei voneinander unabhängigen Bildschirmbereichen.

In Anlehnung an die »Construction Sets« sind den Spielern hier vielfältige Variationsmöglichkeiten in die Hand gegeben worden: So hat man die Wahl zwischen zehn verschiedenen Fahrzeugen. Entscheidend für den Rennablauf ist auch die Motor- und Reifenwahl. Wer seinen Gegner überraschen will, der kann sogar Tellerminen oder Öl in den Kofferraum packen. Verschiedene Straßenverhältnisse (Asphalt, Dreck, Eis) und einstellbare Gravitation sind weitere interessante Aspekte.

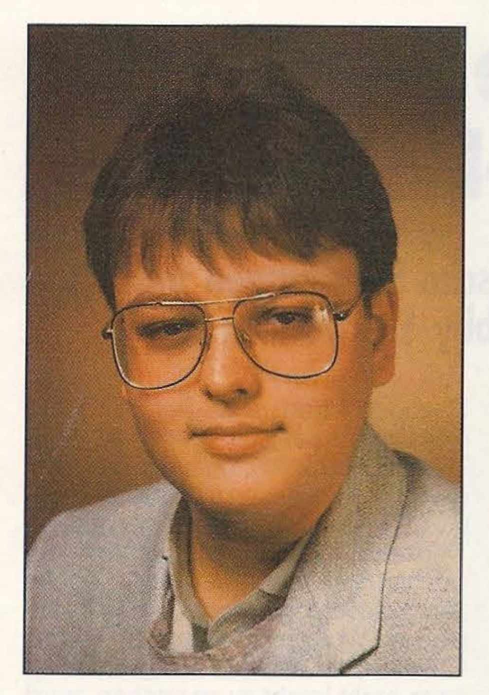
50 verschiedene Rennstrecken werden mitgeliefert. Und wem diese zu langweilig sind, der kann sich über einen eingebauten Editor selber welche basteln.

Ein Nachteil bleibt noch zu erwähnen: Will man nur einmal die Strecke oder die Fahrzeuge wechseln, sind mehrere Minuten Rennpause und Diskettenwechsel angesagt. Insgesamt konnte uns dieses an sich gute Spiel nicht vollkommen überzeugen.

(Boris Schneider/rg)



Titel: Racing Destruction Set
5 7 9 11 13 15
Spielidee
Grafik
Sound
Schwierigkeit ★
Motivation
Besonderheiten: 3D-Effekt auf zwei Bildschirmen
Hersteller: Electronic Arts
Preis: 79 Mark
Bezugsquelle: Ariolasoft, Steinhauserstr. 3, 8000 München 80; Funtastic, Sonnenstr. 9, 8000 München 2
★ Werte einstellbar



Alexander C. Schindowski

Der Autor des Tiny-Forth stellt sich vor

Ich wurde am 11.12.69 in Frankfurt/Main geboren. Schon früh erwachte mein Interesse an Technik, sehr zum Leidwesen meiner Eltern. Von Anfang an hegte ich besonderes Interesse an Elektronik. Doch nachdem ich im Alter von 4 Jahren mehrere Schläge aus heimischen Steckdosen bekommen hatte, beschloß ich, mich auf Schwachstrom zu verlegen.

Nach 4 Jahren Grundschule hatte man mich soweit gebracht, nach dem tieferen Sinn aller Dinge zu suchen. Dies machte sich in einer großen Anzahl demontierter Kassettenrecorder, Radios, ferngesteuerter Autos und Schiffe bemerkbar. Nach mehreren fehlgeschlagenen Versuchen, das Demontierte wieder zusammenzubauen, beschloß ich, mich den Computern zuzuwenden.

An Weihnachten '83 erhörten meine Eltern mein Flehen, und ich fand einen Genie I mit Monitor unter dem Weihnachsbaum, dem im Sommer '84 ein C 64 folgte.

Ich besuchte damals die 9. Klasse des Anna-Schmidt-Gymnasiums in Frankfurt. Bald danach begann ich Pascal und Assembler zu lernen. Auch erwachte mein Interesse an Forth, doch war ich nicht bereit, über 100 Mark für ein entsprechendes Programm auszugeben, da bei mir gerade finanzielle Ebbe herrschte. So entstand das folgende Programm, welches ich in mehreren schlaflosen Nächten entwickelte. Ein weiterer Grund war mein Interesse an der Funktionsweise eines Compilers.

(Alexander C. Schindowski)

Tiny-Forth-Compiler zum Abtippen

Die Programmiersprache Forth ist zur Zeit in aller Munde. Unser Listing des Monats gibt Ihnen die Möglichkeit, Forth einmal praktisch zu erleben.

Forth ist eine der jüngsten Programmiersprachen. Sie wurde 1969 von Charles Moore am National Radio Astronomy Observatory in den USA entwickelt. Der Name der Sprache lautete eigentlich Fourth (das Vierte), aber der IBM-Computer, auf dem Forth entwickelt wurde, ließ nur fünf Buchstaben als Namensangabe zu, so entstand »Forth«.

Forth ist eine der schnellsten Programmiersprachen, die es gibt. Vor allem auf Heimcomputern wird es deshalb gerne eingesetzt. Dazu kommt, daß Forth nicht viel Speicherplatz beansprucht. Die Sprache besteht nicht nur aus einem Compiler, sondern auch aus einem Interpreter; beide arbeiten Hand in Hand.

Die wohl auffälligste Eigenart von Forth ist die Art und Weise, in der Forth rechnet. Es ist die sogenannte »UPN« (Umgekehrte Polnische Notation), auch Postfix-Notation genannt. Sie ist es unter anderem, die Forth die Geschwindigkeit verleiht (10 bis 20mal so schnell wie Basic). Doch was bedeutet UPN?

Ein Beispiel: Sie wollen das Ergebnis von 8 + 5 auf dem Bildschirm ausgeben. In Basic sähe das dann so aus: »PRINT 8 + 5«. In Forth schreibt sich das etwas anders: »5 8 + .«. Scheinbar verwirrend, aber nur auf den ersten Blick. Denn das Prinzip ist einfach. Im Mittelpunkt von Forth steht der Stack (Stapel). Man stelle sich einen Stapel Papier vor, auf den man Blätter obenauf legen kann und auch nur von oben wieder nehmen kann. Das bedeutet, das Blatt, welches Sie zuletzt draufgelegt haben, wird als erstes wieder heruntergenommen. Man nennt dieses System auch »LIFO« (Last In — First Out). Doch wie kann man damit rechnen? Kommen wir wieder zu unserem Beispiel zurück. Der Computer legt als erstes die Zahl 5 auf den Stack. Bild 1 verdeutlicht das Prinzip. Der TOS (Top of Stack) hat jetzt den Wert 5. Dann folgt die »8« nach dem gleichen Verfahren. Darauf addiert der Computer die zwei obersten Zahlen und legt das Ergebnis auf den Stack, dafür ist » + « verantwortlich. Jetzt haben wir zwar das Ergebnis auf dem Stack, können es aber nicht sehen. Für die Ausgabe von 16 Bitzahlen ist der Befehl ».« zuständig. Damit wird immer der jeweilige Wert des TOS ausgegeben.

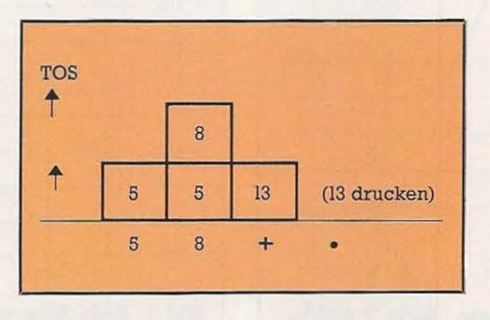


Bild 1. Die Rechnung »5 + 8 = 13« als UPN-Demo

Diese Art des Rechnens mittels UPN mag für Menschen sehr gewöhnungsbedürftig sein, für den Computer ist sie ideal. Doch Forth besitzt neben seiner Geschwindigkeit auch noch weitere Vorteile:

— In Forth können Sie Ihre eigenen Befehle definieren. Es gibt dann in der Benutzung keinen Unterschied mehr zwischen den vordefinierten und Ihren eigenen Befehlen.

— Forth besteht nicht nur aus einem Compiler, sondern auch aus einem Interpreter. Dies gibt Ihnen die Möglichkeit, selbstentwickelte Befehle sofort zu testen.

— Eine sehr hohe Geschwindigkeit beim Compilieren (1 Pass-Compiler)

— Strukturierte Programmierung ohne GOTO

Sie sehen also, Forth ist eine Sprache, mit der zu beschäftigen es sich lohnt.

(Alexander Schindowski/ev)

Netzwerkanalyse — Ein Programm für Hobby-Elektroniker

Dieses Programm simuliert und analysiert elektronische Schaltungen und ist daher ein Muß für jeden ambitionierten Hobby-Elektroniker.

NEWA2, wie das Programm im weiteren genannt wird, ist ein Netzwerkanalyseprogramm für den C 64 mit Diskettenlaufwerk und Simons Basic. Das Programm eignet sich zur Simulation und Analyse passiver und aktiver elektrischer Schaltungen. Es berechnet den Frequenzgang und den Phasenverlauf von Schaltungen, die aus Widerständen, Kondensatoren, Induktivitäten, Transistoren, Operationsverstärkern und allgemeinen Vierpolen bestehen. Das Programm kann somit zum Entwickeln von Verstärkerschaltungen, Filtern und Schaltungen der HF- und Funktechnik herangezogen werden. Neben der Berechnung kann der Frequenzgang durch ein »Bode-Diagramm« grafisch dargestellt werden. Dabei wird die Verstärkung, ausgedrückt durch das logarithmische Maß in dB, über dem logarithmischen Maß der Frequenz aufgetragen. Sowohl die Berechnungen wie aber auch das Bode-Diagramm können auf dem Drucker normiert, das heißt auf einen bestimmten Verstärkungsbetrag bezogen, ausgegeben werden. Zu beachten ist allerdings, daß der benutzte Drucker kompatibel zu den Commodore-Druckern ist beziehungsweise ein kompatibles Interface hat, da die Hardcopy über den Befehl COPY des Simons Basic ausgegeben wird. Die Schaltungsdaten und die Frequenzgänge lassen sich als sequentielles File auf Diskette abspeichern. Dadurch wird die Möglichkeit gegeben, Schaltungen zu einem späteren Zeitpunkt nochmals mit geänderten Werten zu bearbeiten. Die Theorie zu diesem Programm ist zwar nicht leicht zu verstehen, wird aber im folgenden ausführlich behandelt. Anhand von zwei Beispielen, die einen aktiven Bandpaß und einen aktiven Klangregler behandeln, wird versucht, dem Leser dieses Thema näher zu bringen. Wer sich intensiver mit dem Entwurf von Schaltungen beschäftigen möchte, dem sei angeraten, sich das Buch zu diesem Thema »Halbleiter Schaltungstechnik« von U. Tietze und Ch. Schenk, erschienen im Springer-Verlag, zu besorgen.

Die gesamte Bedienung des Programms erfolgt in Menütechnik und ist daher sehr einfach. Fehler werden soweit wie möglich vom Programm abgefangen. Das Eintippen dieses 10 KByte langen Programms ist zwar ein Stückchen Arbeit, aber dafür erspart es jedem ambitionierten Elektronikbastler Stunden des Probierens und Tüftelns beim Schaltungsentwurf. (Hans Haberl/ah)

Lebenslauf

Angefangen hat es ganz ohne Computerunterstützung anno 1960 in Steinhöring (also in Bayern). Schon bald begann ich mich für alles zu interessieren, was irgendwie nach Technik roch. Während andere Mumps oder Röteln hatten, litt ich an Radio-Zerlegeritis, Elektronik-Bastleritis und HiFi-Manie. »Dieser Saustall muß ein anderer werden« hat sich das Christkind wohl gesagt und hat mir anno 1979 einen TRS-80 unter den Christbaum gestellt — natürlich nicht ohne vorherige Absprache mit mir. Irgendwann sind mir dann die 16 KByte und die Grafikauflösung von 48 x 128 Punkten zu wenig geworden und es folgte ein C 64. Natürlich hat sich meine Neigung auch in beruflicher Hinsicht ausgewirkt: Seit 1980 studiere ich Elektrotechnik mit dem Schwerpunkt Datenverarbeitung an der Technischen Universität München. Neben der Computerei ist Musik, besonders klassische, mein Hobby: Brahms, Beethoven, Liszt und Dvořák sind meine Lieblingskomponisten.

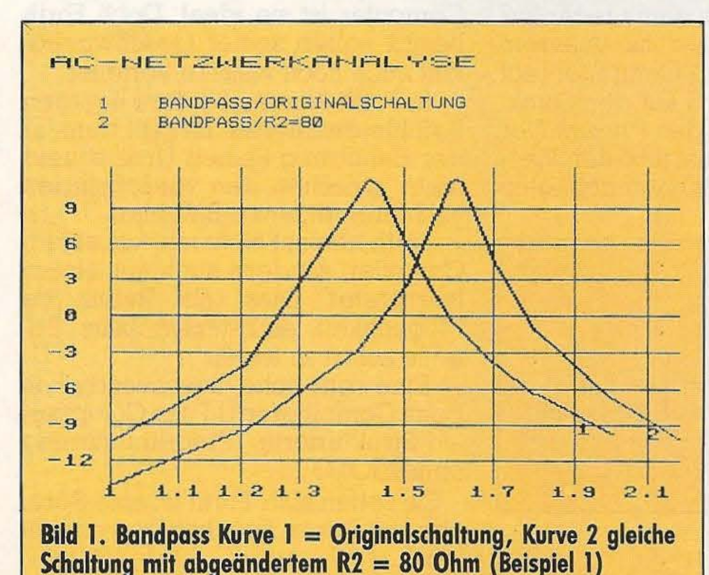


Bild 1. Bandpass Kurve 1 = Originalschaltung, Kurve 2 gleiche Schaltung mit abgeändertem R2 = 80 Ohm (Beispiel 1)

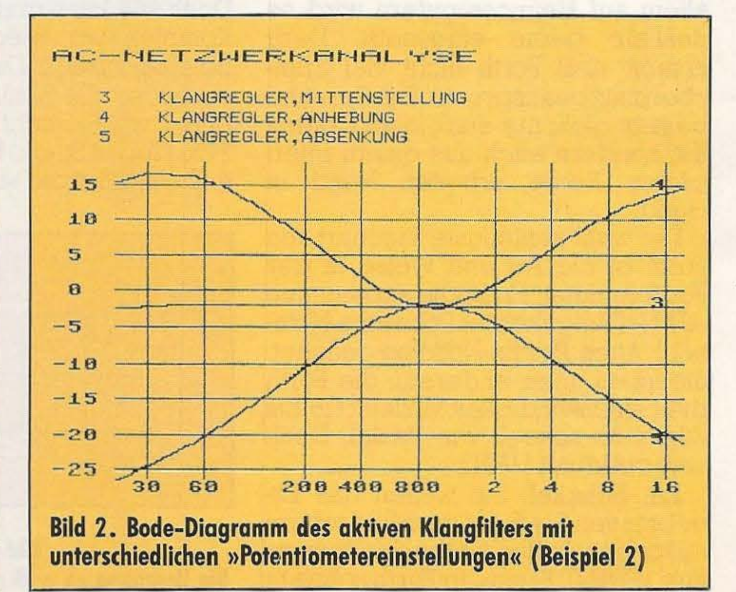


Bild 2. Bode-Diagramm des aktiven Klangfilters mit unterschiedlichen »Potentiometereinstellungen« (Beispiel 2)

Checksummer 64 — Neu

Der Checksummer 64 V3 überprüft jede Basic-Zeile direkt nach der Eingabe, erkennt Fehleingaben und auch Vertauschungen von Zahlen und Ziffern, und erspart deshalb eine aufwendige Fehlersuche.

Der Checksummer 64 V3 ist ein kleines Maschinenprogramm, das Sie sofort unterrichtet, ob Sie die jeweilige Programmzeile korrekt eingegeben haben.

So gehen Sie vor:

1. Programm abtippen und speichern.

2. Starten mit RUN

3. Nach kurzer Zeit sehen Sie am Bildschirm:
CHECKSUMMER 64, CHECKSUMMER AKTIVIERT, AUSSCHALTEN MIT POKE 1,55, ANSCHALTEN MIT POKE 1,53, READY.

4. Anschalten des Checksummer 64 V3 mit POKE 1,53.

5. Test: Geben Sie in einer freien Zeile ein: »1 REM« und drücken die RETURN-Taste. Am Bildschirm oben links sollten Sie die Prüfsumme <63> sehen.

6. Geben Sie ein Listing aus unserem Heft ein. Nach jeder Zeile wird die Zahl, die im Listing in Klammern < > steht, in den Bildschirm eingeblendet. Stimmen die Zahlen nicht überein, so liegt vermutlich ein Eingabefehler vor. **Die Zahl in den Klammern, und auch die Klammern selbst, dürfen beim Abtippen nicht mit eingegeben werden!**

```
 5 PRINT CHR$(14)                                <242>
 10 PRINT"{CLR}"                                  <254>
 20 PRINT"Q*****************************W"        <130>
 30 PRINT"{4DOWN,2SPACE}TEST{SPACE,BLUE,6SP
    ACE}"                                         <022>
 40 PRINT"BBBBBBBBBBBBBBBBBBBBBBBBBBBB"           <108>

© 64'er
```

Bild 1. So könnte ein Teil eines Listings abgedruckt sein. In Zeile 10 müssen Sie nach den Anführungsstrichen die CLEAR/HOME-Taste drücken und nicht die Klammern mit dem Wort CLR. In Zeile 20 drücken Sie nach den Anführungsstrichen die Commodore-Taste und den Buchstaben Q, gefolgt von mehreren SHIFT und Stern-Taste, und zum Schluß die Commodore-Taste und den Buchstaben W. In Zeile 30 ist es viermal die Cursor-nach-unten-Taste, gefolgt von zweimal die Leertaste, dann SHIFT und T und normal EST , zum Schluß noch einmal die Leertaste, die Farbtaste Blau (Control und 7) und sechsmal die Leertaste. Zeile 40 besteht lediglich aus mehreren Grafikzeichen, die mit der Commodore-Taste und B erzeugt werden.

```
5 PRINTCHR$(14)
10 PRINT"[CLR-Zeichen]"
20 PRINT"├──────────────────────────────┤"
30 PRINT"[4 DOWN-Zeichen]  |EST [BLUE-Zeichen]      "
40 PRINT"[Grafikzeichen ...]"
```

Bild 2. Auf dem Bildschirm oder Ihrem Drucker sieht das Listing (Bild 1) so aus.

CTRL steht für Control-Taste, so bedeutet [CTRL-A], daß Sie die Control-Taste und die Taste »A« drücken müssen. Im folgenden steht:

{DOWN}	Taste neben rechtem Shift, Cursor unten
{UP}	Shift-Taste & Taste neben rechtem Shift; Cursor hoch
{CLR}	Shift-Taste & 2. Taste ganz rechts oben
{INST}	Shift-Taste & Taste ganz rechts oben
{HOME}	2. Taste von ganz rechts oben
{DEL}	Taste ganz rechts oben
{RIGHT}	Taste ganz rechts unten
{LEFT}	Shift-Taste & Taste unten rechts
{SPACE}	Leertaste
{F1}	grauer Tastenblock rechts
{F3}	grauer Tastenblock rechts
{F5}	grauer Tastenblock rechts
{F7}	grauer Tastenblock rechts
{F2}	grauer Tastenblock rechts & Shift
{F4}	grauer Tastenblock rechts & Shift
{F6}	grauer Tastenblock rechts & Shift
{F8}	grauer Tastenblock rechts & Shift
{RETURN}	Shift-Taste & Return
{BLACK}	Control-Taste & 1
{WHITE}	Control-Taste & 2
{RED}	Control-Taste & 3
{CYAN}	Control-Taste & 4
{PURPLE}	Control-Taste & 5
{GREEN}	Control-Taste & 6
{BLUE}	Control-Taste & 7
{YELLOW}	Control-Taste & 8
{RVSON}	Control-Taste & 9
{RVOFF}	Control-Taste & 0
{ORANGE}	Commodore-Taste & 1
{BROWN}	Commodore-Taste & 2
{LIG.RED}	Commodore-Taste & 3
{GREY 1}	Commodore-Taste & 4
{GREY 2}	Commodore-Taste & 5
{LIG.GREEN}	Commodore-Taste & 6
{LIG.BLUE}	Commodore-Taste & 7
{GREY 3}	Commodore-Taste & 8

Wenn Sie sich erst einmal an die in Klartext geschriebenen Steuerzeichen gewöhnt haben, werden Sie den Vorteil dieser Schreibweise erkennen. Der zu dem jeweiligen Steuerzeichen gehörende Klartext ist so verfaßt, daß Sie leicht die Taste beziehungsweise die Tastenkombination finden, die Sie drücken müssen.

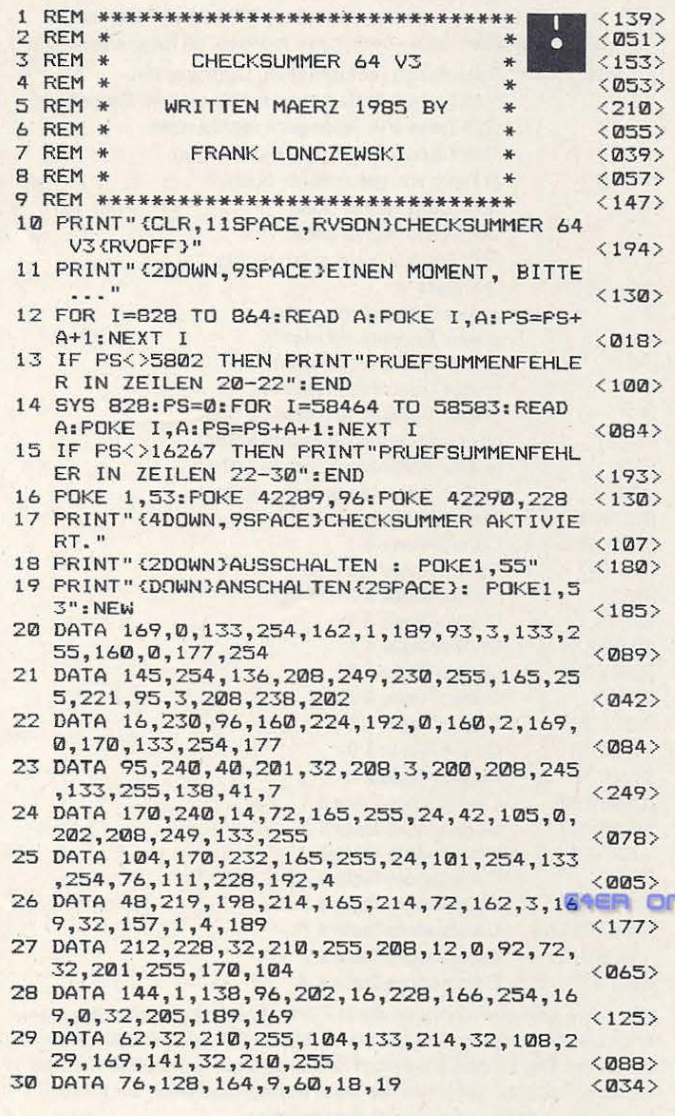
Die Steuerbefehle im Klartext

7. Dieser neue Checksummer 64 V3 bemerkt, im Gegensatz zu den bisherigen, auch Vertauschungen von Zahlen und Buchstaben.

8. Unsere Basic-Listings enthalten keine Steuerzeichen mehr. Diese werden ersetzt durch Klartext und stehen zwischen geschweiften Klammern. Deshalb sind weder die Klammern noch was dazwischen steht, abzutippen, sondern die in Tabelle 1 aufgeführten Tasten zu drücken. Auf Ihrem Bildschirm erhalten Sie dann wieder die entsprechenden Grafikzeichen (siehe Bild 1 und 2).

9. Alle Grafikzeichen werden ebenfalls ersetzt durch unterstrichene oder überstrichene Großbuchstaben. Unterstrichene Buchstaben bedeuten, daß Sie die SHIFT-Taste und den angegebenen Buchstaben drücken müssen, überstrichene jedoch die Commodore-Taste mit dem Buchstaben. Auch hier erhalten Sie am Bildschirm das entsprechende Grafikzeichen und nicht etwa das im Listing erkennbare Zeichen (siehe Bild 1 und 2). (F. Lonczewski/gk)

Hinweis: {13 SPACE} bedeutet 13mal die Leertaste drücken



```
1 REM ******************************  <139>
2 REM *                            *  <051>
3 REM *      CHECKSUMMER 64 V3     *  <153>
4 REM *                            *  <053>
5 REM *    WRITTEN MAERZ 1985 BY   *  <210>
6 REM *                            *  <055>
7 REM *      FRANK LONCZEWSKI      *  <039>
8 REM *                            *  <057>
9 REM ******************************  <147>
10 PRINT"{CLR,11SPACE,RVSON}CHECKSUMMER 64
   V3{RVOFF}"                          <194>
11 PRINT"{2DOWN,9SPACE}EINEN MOMENT, BITTE
   ..."                                <130>
12 FOR I=828 TO 864:READ A:POKE I,A:PS=PS+
   A+1:NEXT I                          <018>
13 IF PS<>5802 THEN PRINT"PRUEFSUMMENFEHLE
   R IN ZEILEN 20-22":END              <100>
14 SYS 828:PS=0:FOR I=58464 TO 58583:READ
   A:POKE I,A:PS=PS+A+1:NEXT I         <084>
15 IF PS<>16267 THEN PRINT"PRUEFSUMMENFEHL
   ER IN ZEILEN 22-30":END             <193>
16 POKE 1,53:POKE 42289,96:POKE 42290,228  <130>
17 PRINT"{4DOWN,9SPACE}CHECKSUMMER AKTIVIE
   RT."                                <107>
18 PRINT"{2DOWN}AUSSCHALTEN : POKE1,55"   <180>
19 PRINT"{DOWN}ANSCHALTEN{2SPACE}: POKE1,5
   3":NEW                              <185>
20 DATA 169,0,133,254,162,1,189,93,3,133,2
   55,160,0,177,254                    <089>
21 DATA 145,254,136,208,249,230,255,165,25
   5,221,95,3,208,238,202              <042>
22 DATA 16,230,96,160,224,192,0,160,2,169,
   0,170,133,254,177                   <084>
23 DATA 95,240,40,201,32,208,3,200,208,245
   ,133,255,138,41,7                   <249>
24 DATA 170,240,14,72,165,255,24,42,105,0,
   202,208,249,133,255                 <078>
25 DATA 104,170,232,165,255,24,101,254,133
   ,254,76,111,228,192,4               <005>
26 DATA 48,219,198,214,165,214,72,162,3,16
   9,32,157,1,4,189                    <177>
27 DATA 212,228,32,210,255,208,12,0,92,72,
   32,201,255,170,104                  <065>
28 DATA 144,1,138,96,202,16,228,166,254,16
   9,0,32,205,189,169                  <125>
29 DATA 62,32,210,255,104,133,214,32,108,2
   29,169,141,32,210,255               <088>
30 DATA 76,128,164,9,60,18,19          <034>
```


Dieser neue Checksummer 64 V3 erkennt auch Vertauschungen von Zahlen.

MSE – Abtippen leicht gemacht

Ähnlich wie der »Checksummer« ist auch der MSE ein Hilfsmittel bei der Eingabe von Listings, diesmal jedoch bei reinen Maschinensprache-Programmen.

Im Gegensatz zum »Checksummer« aber ist die Eingabe nicht ohne den MSE möglich. Der MSE verringert die Tipparbeit um ein Drittel und schließt Fehleingaben vollkommen aus. Außerdem können Sie die Werte blind eingeben, ohne andauernd auf den Bildschirm schauen zu müssen. Dies wird durch akustische Meldungen realisiert.

MSE ist ein Maschinenspracheditor, mit dem ein Vertippen ausgeschlossen ist. Eine abgetippte Zeile wird nur angenommen, wenn sie richtig ist. Eine Checksumme am Ende jeder Zeile prüft, ob die richtigen Werte in der richtigen Zeile an der richtigen Stelle stehen. Wenn nicht, ertönt ein Warnsignal, und man beseitigt den Fehler.

War die Zeile korrekt, erklingt ein Gong, und die nächste Zeilennummer wird ausgegeben. Damit ist also auch »blindes« Eintippen möglich; Sie können sich voll auf den Text konzentrieren.

So arbeitet man mit MSE

Laden und starten Sie MSE. Zuerst wird der Programmname und die Start- und Endadresse erfragt. **Diese Angaben entnehmen Sie dem Kopf des jeweiligen abgedruckten Listings.** MSE meldet sich dann mit der Zeilennummer der ersten Zeile. Wenn Sie die Zeile richtig eingegeben haben, erscheint die nächste Zeilennummer und so weiter bis zum Ende. Zum Schluß wird das fertige Programm mit »CTRL-S« auf Diskette oder Kassette abgespeichert. Dazu sind keine weiteren Angaben mehr erforderlich. Das Programm kann dann ganz normal wieder geladen und gestartet werden. Wenn Sie nicht alles auf einmal tippen wollen, können Sie jederzeit unterbrechen und den eingetippten Teil mit »CTRL-S« abspeichern. Wollen Sie weiterarbeiten, laden und starten Sie MSE wieder.

Geben Sie auf die Frage nach der Startadresse aber jetzt »CTRL-L« ein, um Ihr Teilprogramm zu laden. Jetzt können Sie mit »CTRL-N« die Adresse eingeben, an der Sie weitertippen müssen. Wenn Sie sich nicht gemerkt haben, wie weit Sie gekommen sind, geben Sie nach dem Laden »CTRL-M« ein.

Auf die Frage nach der Startadresse antworten Sie mit der Anfangsadresse, die links in der Kopfzeile auf dem Bildschirm steht. Nun wird Ihr Programm aufgelistet. Mit »SPACE« wird das Listen fortgesetzt, mit »STOP« abgebrochen. Das Ende Ihres Programmteils erkennen Sie sehr einfach daran, daß nur noch der Wert »AA« in der Zeile steht. Die Adresse dieser Zeile müssen Sie anschließend mit »CTRL-N« eingeben. Das Programm ist nur mit »STOP/RESTORE« zu verlassen. Speichern Sie aber vorher unbedingt immer Ihren Text ab.

Hinweise zum Abtippen

Vor dem Abtippen oder späteren Wiederladen des MSE-**Laders** müssen Sie unbedingt folgende Zeile eingeben:

POKE 43,1: POKE 44,32: POKE 8192,0: NEW

Starten Sie das Programm mit RUN. Fehlerhafte Zeilen werden angezeigt und müssen korrigiert werden, bis der Lader zum »READY« durchläuft. Jetzt müssen Sie das fertige MSE-Programm abspeichern. Dazu brauchen Sie nur »RETURN« zu drücken, weil die erforderlichen Angaben schon auf dem Bildschirm stehen. (Kassettenbesitzer müssen in Zeile 343 die letzte Zahl in »1« abändern.) Ab jetzt können Sie »MSE V1.0« direkt, also ohne den DATA-Lader, benutzen. MSE V1.0 wird ganz normal mit »,8« geladen (keine POKEs notwendig).

(N. Mann / D. Weineck / gk)

Das Listing

Der Checksummer und der MSE befinden sich auf jeder Leserservice-Diskette. Gegen Einsendung eines frankierten und an Sie selbst adressierten Briefumschlags (Größe DIN C5 = 0,80 Mark, DIN C4 = 1,10 Mark Porto) schicken wir Ihnen gerne das MSE-Listing zu. Es ist jedoch ebenfalls in den 64'er-Ausgaben 1/85 bis 6/85 abgedruckt.

MSE-Befehle:

DEL	löscht die letzte Eingabe.
CTRL-S	speichert das eingetippte Programm ab.
CTRL-L	lädt ein Programm. Start- und Endadresse werden automatisch ermittelt.
CTRL-M	listet den Speicherinhalt. Abbruch mit STOP-Taste, weiter mit Leertaste.
CTRL-N	erlaubt die Eingabe einer neuen Adresse zum Weitertippen.
CTRL-P	gibt ein MSE-Listing auf dem Drucker aus.



Netzwerkanalyse — Die Hilfe für Hobby-Elektroniker

Durch dieses Programm erspart sich jeder Elektronik-Bastler Stunden des Probierens und Tüftelns beim Entwurf frequenzkritischer Schaltungen.

Das Programm ist in Simons Basic geschrieben. Sie müssen also zuerst diese Erweiterung laden, bevor Sie »NEWEA2« (Listing 1) eintippen.

Wenn Sie damit fertig sind, wollen wir gleich voll einsteigen und die Bedienung anhand eines Beispiels erklären.

Nach dem Initialisieren meldet sich NEWEA2 mit dem Hauptmenü. Darunter blinkt ein gelber Cursor, was heißt, daß die angeforderte Eingabe nur aus einem Zeichen besteht und nicht mit RETURN abgeschlossen werden muß. Bei einem grauen Cursor dagegen muß die Eingabe mit RETURN abgeschlossen werden, Sie können auch nur ein RETURN eingeben, wenn der alte Wert vor dem Fragezeichen steht und übernommen werden soll. Werden mehrere Eingaben auf einmal angefordert, so müssen diese durch Kommata getrennt werden.

Ein Beispiel

Netzwerkeingabe

Geben Sie nun »N« für Netzwerkeingabe ein. Wir wollen nämlich den in Bild 1 gezeigten aktiven Bandpaß als erstes Beispiel analysieren. Das Programm fragt nun nach der Anzahl der Knoten. Ein Knoten ist jedes zugängliche Potential in einer Schaltung, also die Punkte, die Sie auch mit einem Meßgerät erreichen können. Da der Masseknoten nicht mitgezählt wird, sind es in unserem Beispiel vier Knoten. Der Knoten zwischen Quelle U0 und Quellwiderstand R0 zählt ebenfalls nicht, da er nicht zur untersuchten Schaltung gehört. Nun geben Sie auf die Fragen des Programms die Anzahlen der Elemente ein: drei Widerstände, zwei Kondensatoren und ein Operationsverstärker, sonst überall 0.

Nun kommt — wie immer nach einem Block von Eingaben — die Frage »Korrektur?« Geben Sie hier ein »J« für Ja, so können Sie diesen Eingabeblock wiederholen, um eventuelle Fehleingaben zu korrigieren. Bei jedem anderen Zeichen fährt das Programm normal fort. Apropos Fehleingaben: Unerlaubte Eingaben werden fast alle erkannt und durch Wiederholen der Frage oder des ganzen Blocks quittiert. »Fast« sage ich deshalb, weil es auch »intelligente« Fehleingaben gibt, die das Programm nicht sofort erkennt, wie zum Beispiel ein Knoten, an dem kein Element liegt, ein Kurzschluß am Ausgang oder galvanisch getrennte Teilschaltungen (die Schaltung muß zusammenhängen, da NEWEA2 ein gemeinsames Bezugspotential benötigt). Solche Fehler werden erst bei der Berechnung erkannt.

Nun möchte das Programm wissen, an welchen Knoten die Elemente anliegen. Bei der Numerierung der Knoten müssen Sie zwei Dinge beachten:

1. Der Masseknoten besitzt die Nummer 0.
2. Der Eingangsknoten besitzt die Nummer 1. An diesen Knoten legt das Programm automatisch die Eingangsspannungsquelle, Sie müssen diese also nicht explizit angeben.

Die Numerierung der übrigen Knoten sowie der Bauelemente (außer R0) ist beliebig. Somit dürfte die Eingabe des Netzwerkes kein Problem mehr sein. Außerdem können Sie im Beispielausdruck sehen, wie es geht. Bei mehr als zweipoligen Elementen wird die Reihenfolge der Knoten — CBE beim Transistor und ±A beim Operationsverstärker — mit angezeigt. Der zuletzt erfragte Knoten — der Ausgangsknoten — ist derjenige Knoten, dessen Potential als Ausgangsspannung betrachtet wird. Das muß nicht unbedingt der tatsächliche Ausgang der Schaltung sein, sondern Sie können sich beliebige Spannungen innerhalb der Schaltung ansehen. In unserem Beispiel interessiert uns jedoch der richtige Ausgang, also Knoten 4.

Als nächstes müssen wir die Elementewerte eingeben, und zwar in Grundeinheiten, also **Ohm, Farad, Henry, Hertz, Ampere** etc. Beim Operationsverstärker wird noch genauer formuliert, was eingegeben werden soll: die Leerlaufspannungsverstärkung V0 und die Transitfrequenz FT, hier mit 100000 beziehungsweise 1 MHz angenommen.

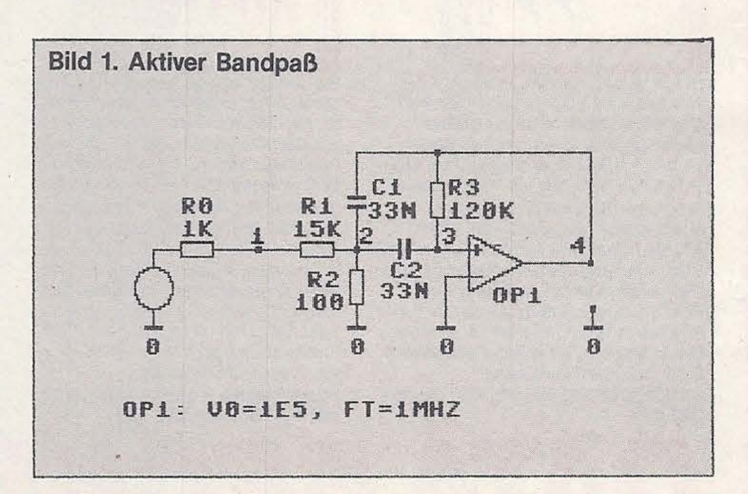
Dann rechnet das Programm zum ersten Mal, es stellt nämlich die sogenannte Knotenleitwertmatrix Yn sowie die Strukturmatrix auf. Während das Programm arbeitet, ist es nicht ansprechbar und macht dies durch das Wörtchen »busy« klar. Danach können Sie das Netzwerk noch auf den Drucker ausgeben lassen. Sollten Sie hier ein »J« getippt haben, der Drucker aber gerade nicht gewillt sein zu drucken, so bricht das Programm zwar mit einem »Device not present« ab, kann aber durch Drücken der F1-Taste ohne Datenverlust fortgesetzt werden.

Frequenzgangberechnung

Jetzt sind wir wieder im Hauptmenü und geben »F« für Frequenzgangberechnung ein. Nun müssen wir ein »Frequenzfile« angeben, in dem wir den Frequenzgang speichern wollen. Diese Nummer darf zwischen 1 und 9 liegen und dient dazu, später wieder Bezug auf diesen Frequenzgang nehmen zu können, zum Beispiel um ihn grafisch darzustellen oder auf Diskette zu speichern. Geben Sie hier eine 0 ein, so kehrt das Programm ins Hauptmenü zurück. Wie Sie später sehen werden, dient die 0 oft dazu, eine Eingabe abzubrechen. Zusätzlich zur Nummer können Sie nun noch eine Notiz eingeben, die später immer zusammen mit der Nummer angezeigt wird. Man braucht sich also nicht zu merken, welche Nummer ein bestimmter Frequenzgang hat. In unserem Beispiel könnte die Notiz zum Beispiel lauten:

»Bandpaß, Originalschaltung« (Wenn der Text Kommata oder Doppelpunkte enthält, muß er in Anführungszeichen gesetzt werden. Das gleiche gilt beim Speichern, wenn Sie einem Filenamen einen Klammeraffen vorausstellen, um ein bereits existierendes File zu überschreiben).

Bild 1. Aktiver Bandpaß



Jetzt wird's ernst: Geben Sie nun die erste Frequenz (im Bereich zwischen 1 000 und 2 000 Hz) ein! Der Bandpaß hat seine Resonanzfrequenz im Bereich zwischen 1 und 2 kHz. Suchen Sie diese doch einmal, ohne im Beispielausdruck nachzuschauen. In ein Frequenzgangfile passen maximal 20 Frequenzen. Wollen Sie die Eingabe schon früher abbrechen, so geben Sie eine 0 ein. Hier gleich noch ein Tip zum Suchen von Resonanzfrequenzen: Schauen Sie nicht nur auf den Betrag, sondern auch auf die Phase. Bei Resonanzfrequenzen ist dies meist ein ganzzahliges Vielfaches von 90 Grad, in unserem Beispiel 180 Grad.

Nach Abbruch der Frequenzgangberechnung stellt das Programm die Frage, ob Sie das File mit einem anderen vergleichen wollen (#), ob Sie es normieren (N) oder auf den Drucker (D) ausgeben wollen. Drücken Sie hier erst mal eine beliebige andere Taste (zum Beispiel RETURN oder Space), um ins Hauptmenü zurückzukommen. Da Sie die Frequenzen beim Suchen des Maximums bestimmt durcheinander eingegeben haben, wollen wir jetzt in Ruhe den sortierten Frequenzgang betrachten: Geben Sie »A« ein und dann die Nummer, die Sie für Ihr Frequenzfile gewählt haben. Die Frage »Notiz« beantworten Sie mit RETURN, damit die alte Notiz übernommen wird. Jetzt wird der ganze Frequenzgang fein säuberlich sortiert ausgegeben.

Nun wollen wir an unserer Schaltung ein bißchen »herumbiegen« und sehen, was passiert. Drücken Sie dazu erst wieder eine Taste, um ins Hauptmenü zu kommen und dann ein »E« für Elementeingabe. Dadurch sparen wir uns das ganze »Durchtippen« der Netzwerkeingabe und kommen gleich zu dem Eingabeblock für die Elementwerte. Ändern Sie den Wert für R2 von 100 nach 80 Ohm und lassen Sie die übrigen Werte gleich. Wählen Sie dann wieder den Menüpunkt »F« an und geben eine neue Frequenzfilenummer an, zum Beispiel die 2, wenn Sie vorher die 1 belegt hatten. Damit wir nun nicht alle Frequenzen noch einmal eintippen müssen, bietet uns das Programm die Möglichkeit, Frequenzen von einem bereits belegten Frequenzfile zu übernehmen. Geben Sie dazu auf diese Frage die Nummer Ihres ersten Files, also zum Beispiel 1 ein. Wenn Sie auf die nächste Frage nun ein »J« geben, dann wird der neue Frequenzgang ohne Zwischenfragen mit den Frequenzen des alten Files berechnet, geben Sie dagegen kein »N« so fragt das Programm bei jeder Frequenz nach und Sie können diese durch RETURN übernehmen oder aber durch eine andere ersetzen. Probieren Sie es erst einmal mit »J«. Sie werden dann jedoch feststellen, daß sich das Maximum verschoben hat und dort, wo es vorher war, sich jetzt viele unnütze Frequenzen herumräkeln, während das neue Maximum nicht genau erfaßt ist. Wir wollen deshalb einen zweiten Versuch machen: Geben Sie vom Hauptmenü aus wieder »F« ein und anschließend wieder Filenummer 2. Auf die Frage »Frequenzen von File« brauchen Sie diesmal nur mit RETURN zu antworten, da die Frequenzen von File 1 ja schon vorher in File 2 kopiert wurden. Auf die Frage »Ohne Änderung« geben Sie jedoch diesmal ein »N«. Das Programm hält nun bei jeder Frequenz an und Sie können diese durch RETURN übernehmen oder eine andere eingeben.

Wenn Sie die Frequenzgangberechnung abgeschlossen haben, gehen Sie nicht gleich ins Hauptmenü zurück, sondern geben Sie auf die Frage »File #, Normierung oder Drucker« die Nummer Ihres ersten Files, also 1 ein. Das zuletzt berechnete File wird nun mit File 1 verglichen, wobei nur gleiche Frequenzen nebeneinanderstehen. Außerdem erfolgt die Ausgabe nur solange, wie sich die Files überdecken. Nun können Sie gleich noch den Punkt »Normierung« ausprobieren. Damit können Sie den Frequenzgang bei einer bestimmten Bezugsfrequenz auf einen bestimmten dB-Wert normieren, also zum Beispiel 0 dB bei der Resonanzfrequenz (die Bezugsfrequenz muß also im Frequenzgang enthalten sein).

Frequenzgangdiagramm

Da die Zahlenreihen für Frequenzgänge nicht unbedingt anschaulich sind, kann NEWEA2 das ganze auch grafisch: Geben Sie dazu »D« vom Hauptmenü aus. Da mehrere Frequenzgänge in einem Diagramm dargestellt werden können, fragt das Programm hier nach den Nummern für das 1. (zu zeichnende) File, das 2. und so weiter. Wenn Sie nicht alle belegten Files auf einmal zeichnen wollen, können Sie auch hier die Eingabe durch eine 0 beenden. Zum Diagramm selbst: Auf der Y-Achse sind die dB-Werte aufgetragen, auf der X-Achse in logarithmischem Maßstab die Frequenz in Hz, KHz oder MHz. Diese scheinbar inkonsequente Benennung wurde gewählt, um zu lange Zahlen zu vermeiden. Durch Drücken einer beliebigen Taste kommen Sie wieder in den Normalmodus zurück und können dort durch ein »J« auf die nächste Frage das Diagramm ausdrucken lassen. Dazu brauchen Sie allerdings einen Commodore-Drucker VC 1515, VC 1525, MPS 801 oder einen anderen (zum Beispiel Epson) mit Commodore-kompatiblem Interface, da hier der COPY-Befehl des Simons Basic benutzt wird. Für andere Drucker können Sie jedoch das Programm anpassen, was sehr leicht ist, da alle Druckerbefehle in den Zeilen ab Nummer 7000 zusammengefaßt sind.

Speichern und Laden von Daten

Sowohl die Schaltungsdaten als auch die Frequenzfiles können auf Diskette gespeichert werden. Dazu wählen Sie im Hauptmenü den Punkt »S«. Danach geben Sie auf die Fragen »Schaltung speichern« beziehungsweise »Frequenzfiles speichern« je nach Bedarf ein »J« oder ein anderes Zeichen ein. Bei der Eingabe des Filenamens haben Sie durch Eingabe von »$« die Möglichkeit, sich alle sequentiellen Files auf einer Diskette anzeigen zu lassen. Beim Filenamen empfehle ich Ihnen, Schaltungs- und Frequenzgangfiles entsprechend zu markieren, zum Beispiel durch ein vorangestelltes S/.. beziehungsweise F/... Die Schaltungsdaten können Sie durch eine Notiz ergänzen, die beim Laden wieder ausgegeben wird, bei den Frequenzfiles wird natürlich die vorher eingegebene Notiz mit abgespeichert. Die Eingabe der Frequenzfilenummern geschieht genau so wie beim Diagramm, also am Ende der Eingabe durch 0.

Zum Laden ist noch folgendes zu sagen. Da im Speicher nur immer eine Schaltung Platz hat, wird beim Laden einer neuen Schaltung die alte gelöscht. Ganz anders dagegen bei den Frequenzfiles: Hier werden die Files von Diskette nur in noch freie Frequenzfiles geladen. Das hat folgende Konsequenzen:

1. Es gehen grundsätzlich keine im Computer gespeicherten Daten verloren.
2. Frequenzfiles haben beim Laden in der Regel nicht die selben Nummern wie beim Speichern. Identifizieren kann man sie ja anhand der Notizen.
3. Wenn nicht genug freie Files im Computer vorhanden sind, werden nur so viele Files geladen, wie noch Platz ist. Dies wird durch eine Meldung beim Laden (zum Beispiel »nur 2 von 3«) mitgeteilt. Um im Computer Platz zu schaffen, muß man daher eventuell Files löschen. Dies geschieht ganz einfach durch Anwählen von »F« und der zu löschenden Filenummer und der Eingabe »0« als erste Frequenz.

Zur Theorie

Wenn Sie mit dem Programm »gespielt« haben und den Umgang damit beherrschen, dann wird es Zeit für die Theorie. Keine Angst, ich will Ihnen hier nicht die Grundlagen der Netzwerksynthese beibringen, aber einige Punkte müssen Sie zur Arbeit mit dem Programm wissen.

Lineare Schaltungen

Vielleicht haben Sie sich schon gefragt, warum bei den Elementen keine Dioden vorhanden sind. Der Grund dafür ist, daß

NEWEA2 nur lineare Schaltungen untersuchen kann, das heißt der Zusammenhang von Strom und Spannung an jedem Element der Schaltung muß von der Form U = I x Konst. sein. Der Wert eines Widerstandes zum Beispiel muß immer gleich sein, ob an diesem Widerstand nun 1 V oder 10 V anliegen. Bei der Diode ist das aber nicht der Fall, ihr Widerstand ist abhängig von der anliegenden Spannung. Natürlich gibt es Programme, die auch nichtlineare Schaltungen untersuchen können, das sind sogenannte Transientenanalyse-Programme. Jedoch ist der Rechenaufwand dafür für den C 64 zu groß.

Wir müssen uns also auf lineare Schaltungen beschränken. Aber Transistor und Operationsverstärker sind auch nichtlineare Elemente! Wir können sie aber dennoch verwenden, wenn wir darauf achten, daß diese Elemente in den zu untersuchenden Schaltungen im linearen Bereich betrieben werden, denn das Programm sieht diese Elemente grundsätzlich als linear an.

Beim Operationsverstärker heißt das, daß die Ausgangsspannung, die durch die Versorgungsspannung gesetzten Grenzen nicht überschreiten darf, beim Transistor wird vorausgesetzt, daß dieser in einem vernünftig gewählten Arbeitspunkt betrieben wird. Natürlich gibt es auch Schaltungen, die die nichtlinearen Eigenschaften dieser Bauelemente ausnutzen, zum Beispiel Transistor als Schalter oder als Komparator. Solche Schaltungen können mit NEWEA2 nicht untersucht werden. Die Diode könnte man auch in einem Arbeitspunkt linearisieren und dann als normalen Widerstand darstellen. In der Praxis werden jedoch fast immer die nichtlinearen Eigenschaften einer Diode ausgenutzt, weshalb diese Darstellung kaum vorkommen wird.

Das Ersatzschaltbild (ESB)

Als Grundelemente zum Aufbau der zu untersuchenden Schaltungen stehen nur Widerstände, Kapazitäten, Induktivitäten sowie konstante und gesteuerte Strom- und Spannungsquellen zur Verfügung. Andere Elemente müssen aus diesen Elementen in bestimmten Anordnungen — den sogenannten Ersatzschaltungen — aufgebaut werden. Das einfachste Ersatzschaltbild ist das einer Spule: Da eine Spule eben nicht nur eine Induktivität, sondern auch einen ohmschen Widerstand besitzt, sollte sie — vor allem bei HF-Anwendungen — als Serienschaltung einer Induktivität und eines Widerstandes eingegeben werden. Der Widerstand hat dabei den Wert:

$$R = \frac{2\pi f\ L}{Q}$$

wobei Q die Güte der Spule ist und f die Frequenz, bei der diese Güte gemessen wurde. Theoretisch besitzt die Spule auch noch eine Kapazität, ja sogar ein Widerstand hat Kapazitäten und Induktivitäten (Anschlußdrähte!), jedoch sind diese Effekte vernachlässigbar. Lediglich bei extremen HF-Anwendungen haben diese »parasitären« Elemente spürbare Einflüsse. Wenn man jedoch in solchen Frequenzbereichen arbeitet, kann man Schaltungssimulationen und Berechnungen sowieso vergessen, da hilft nur noch eins: Basteln.

Weitere Ersatzschaltbilder können natürlich beliebig eingegeben werden, diejenigen für Transistor und Operationsverstärker sind im Programm bereits vorhanden und werden in den nächsten Abschnitten noch besprochen.

Der Transistor

Das Ersatzschaltbild für den Transistor ist in Bild 2 zu sehen. Es ist ein Pi-Ersatzschaltbild mit fünf Elementen und in dieser Form im Programm bereits gespeichert. Es müssen lediglich drei Daten eingegeben werden:

IC: Kollektorstrom im Arbeitspunkt.

BO: Kurzschlußstromverstärkung (zirka 20....300).

FT: Transitfrequenz (einige MHz bis GHz).

Die thermische Spannung Ut sowie der Basisweitenmodulationsfaktor n (eta) werden vom Programm mit 26 mV beziehungsweise 5 x 10^{-4} angenommen. Sollten diese Werte einmal nicht zutreffen, so kann der Transistor als Vierpol eingegeben werden. Natürlich kann auch das angegebene Transistor-Ersatzschaltbild erweitert werden. Sinnvoll ist allerdings nur, den Basisbahnwiderstand als Vorwiderstand in die Basisleitung zu schalten. Der hat zwar nur zirka 10 bis 100 Ohm, bildet aber einen frequenzabhängigen Spannungsteiler mit C_{BE}. Er ist jedoch nur bei HF-Anwendungen und niederohmiger Ansteuerung des Transistors (oder Basisschaltung) sinnvoll.

Wie schon in einem früheren Abschnitt gesagt, muß der Transistor in einem sinnvoll gewählten Arbeitspunkt betrieben werden. Charakterisiert ist dieser Punkt durch den Kollektorstrom. In einem der noch folgenden Beispiele werden Sie sehen, daß die Eingabe dieser Werte jedoch meist recht unkritisch ist.

Der Operationsverstärker

Der Operationsverstärker ist im Programm als spannungsgesteuerte Spannungsquelle mit frequenzabhängiger Verstärkung V dargestellt. Ersatzschaltbild und die Formel für V sind in Bild 3 zu sehen. Die Frequenzabhängigkeit ist durchaus wichtig, viele Dimensionierungstabellen oder Berechnungsschemen betrachten den Operationsverstärker als ideal und führen daher zu ungenauen Ergebnissen. Der Eingangswiderstand des Operationsverstärkers ist unendlich, der Ausgangswiderstand 0, was natürlich durch Hinzufügen von Elementen geändert werden kann. In der Regel kann man den Operationsverstärker jedoch ohne Ergänzungen übernehmen und erhält genügend genaue Ergebnisse. Wie schon erwähnt, darf die Ausgangsspannung den Versorgungsspannungsbereich nicht verlassen.

Als Daten für den Operationsverstärker muß nur seine Leerlaufspannungsverstärkung für tiefe Frequenzen V0 (zirka 10^5) sowie seine Transitfrequenz FT (einige MHz) eingegeben werden.

Der Vierpol

Ein Vierpol (oder richtiger: Zweitor) ist eine »Black Box« mit den Eingangsklemmen 1 und 1' und den Ausgangsklemmen 2 und 2' (siehe Bild 4). Seine Übertragungseigenschaften werden durch eine Matrix, die sogenannte Leitwertmatrix Y beschrieben. Diese Matrix besteht aus den Größen Y11, Y12, Y21 und Y22. Die Größen berechnen sich folgendermaßen:

Y11 = I1/U1 bei U2 = 0 (sekundärer Kurzschluß)
Y12 = I1/U2 bei U1 = 0 (primärer Kurzschluß
Y21 = I2/U1 bei U2 = 0
Y22 = I2/U2 bei U1 = 0

Außerdem lassen sich alle Ys in drei Teile zerlegen: Den ohmschen, den kapazitiven und den induktiven. Formelmäßig ausgedrückt:

$$Yij = 1/R + j\omega C + 1/j\omega l$$

wobei w (Omega = 2 x pi x f = Kreisfrequenz und j eine komplexe Einheit ist.

Sie müssen für alle 4 Ys jeweils die Komponenten 1/R, C und 1/L eingeben, also insgesamt 12 Werte. Für einige Vierpole

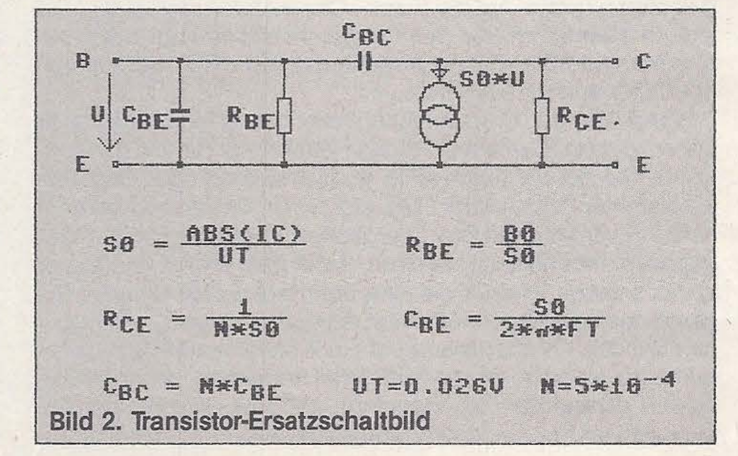


Bild 2. Transistor-Ersatzschaltbild

folgen noch Beispiele, dort werden Sie sehen, daß die Sache nur halb so schlimm ist, wie sie hier aussieht.

Das Wechselstrom-Ersatzschaltbild

Wir haben nun gesehen, wie sich verschiedene Elemente, teils dargestellt durch Ersatzschaltbilder, in das Programm eingeben lassen. Meist ist es jedoch nötig, die gesamte zu untersuchende Schaltung umzuformen. Da wir nämlich grundsätzlich nur das Wechselstromverhalten (Frequenzgang) einer Schaltung untersuchen, interessiert sich das Programm überhaupt nicht für Bauteile, die nur das Gleichstromverhalten der Schaltung etwas angehen. Das offensichtlichste Beispiel dafür sind die Versorgungsspannungen: Wechselspannungsmäßig gesehen ist eine Gleichspannungsquelle ein Kurzschluß und kann deshalb auch als solcher dargestellt werden. Das heißt die Versorgungsspannungsknoten erhalten wie die Masse die Nummer 0 und dürfen bei der Bestimmung der Knotenzahl nicht mitgezählt werden.

Auch Siebelkos sowie Koppelkondensatoren, die so groß sind, daß sie im betrachteten Frequenzbereich keinen Einfluß mehr haben, können durch Kurzschlüsse ersetzt werden.

Man kann dadurch Knoten und somit Rechenzeit sparen. Apropos Knoten sparen: Vielleicht ist Ihnen aufgefallen, daß wir in unserem ersten Beispiel, dem Bandpaß, die Widerstände R0 und R1 zusammenfassen können. Auch dadurch spart man einen Knoten und damit Rechenzeit, allerdings wollte ich nicht gleich am Anfang mit solchen Tricks kommen, um Sie nicht zu verwirren.

Im folgenden Kapitel wird es endlich wieder praktisch, es kommen die versprochenen Beispiele.

Beispiele, Tips und Tricks

Aktiver Klangregler. Bild 5 zeigt einen aktiven Klangregler, der wegen seiner Regelcharakteristik auch oft als »Kuhschwanzregler« bezeichnet wird: Die Enden des Frequenzganges lassen sich nämlich wie ein Kuhschwanz auf- und abschwenken.

Sehen wir uns nun die Schaltung näher an: Bei der Knotennumerierung werde ich mich auf die im Bild angegebene beziehen, eine beliebige andere Reihenfolge wäre natürlich ebenso möglich. Beachten Sie, daß die 28-V-Versorgungsspannung keine eigene Knotennummer hat, sie bekommt wie die Masse die Nummer 0. Die beiden Potentiometer stellt man durch je zwei Widerstände dar, deren Summe den Gesamtwert von 100 kΩ ergeben muß. Durch Variieren der beiden Werte läßt sich das Potentiometer »verstellen«, die Randpositionen lassen sich auch dadurch erreichen, daß man den Abgriff auf Knoten 2 oder 4 beziehungsweise 6 oder 7 umlegt (Widerstände mit dem Wert 0 können nicht eingegeben werden!).

Um Knoten zu sparen, hätte man, wie früher schon erwähnt, auch die großen Kondensatoren durch Kurzschlüsse ersetzen können. Allerdings haben diese — vor allem der 5 µF-Kondensator — im Bereich um 20 Hz schon einen spürbaren Einfluß. Experimentieren Sie doch hier ein wenig!

Der Kollektorstrom des Transistors im Arbeitspunkt läßt sich in diesem Beispiel sehr leicht ermitteln, da die Gleichspannungen am Kollektor und Emitter gegeben sind. Am Kollektorwiderstand (3.9 kΩ) fallen 8 V ab, das ergibt einen Kollektorstrom von IC = 8 V/3 900 Ohm = 2 mA. Da der Emitterstrom ungefähr gleich dem Kollektorstrom ist, wäre es auch damit gegangen: IC = IE = 2 V/1 000 Ohm = 2 mA. In zwei der Beispielfiles wurde der Kollektorstrom um den Faktor 10 (!) nach oben und unten verändert, die Ergebnisse weichen dennoch nicht gewaltig vom richtigen Ergebnis ab (siehe Frequenzgangdiagramm dazu). Sie sehen daraus, daß die Eingabe dieses Wertes nicht sehr kritisch ist. Das gleiche gilt auch für die anderen beiden Transistorparameter B0 und FT. Ich empfehle Ihnen, hier im Einzelfall ein wenig herumzuprobieren, wie empfindlich eine Schaltung auf diese Eingabeparameter reagiert. In der Regel genügt für die Bestimmung dieser Werte die berühmte »Pi-mal-daumen«-Methode.

Man kann jedoch bereits aus dem Beispiel einige Zusammenhänge erkennen:

1. B0 hat nur dann einen Einfluß, wenn der Transistor wenig rückgekoppelt ist, also verstärken soll.

2. IC ist hauptverantwortlich für die Verstärkung. Bei zunehmender Verstärkung wird jedoch auch der Eingangswiderstand des Transistors verkleinert. Diese beiden Effekte können sich kompensieren (die Kurven für 0.2 mA und 20 mA sind fast gleich!). Mit zunehmendem IC wird auch die Grenzfrequenz des Transistors herabgesetzt.

3. FT schließlich ist nur dann interessant, wenn der Transistor an seiner Frequenzgrenze betrieben wird.

Wie Sie gesehen haben, kann man aus der Beschaltung des Transistors seinen Kollektorstrom abschätzen. Natürlich muß auch hier, ebenso wie beim Operationsverstärker sichergestellt sein, daß der Transistor nur im aktiven Bereich betrieben wird. Das Programm erkennt eine Übersteuerung des Transistors nicht.

Übertragungsmaß

Das Übertragungsmaß ist das Verhältnis von Ausgangs- zu Eingangsspannung, also UA/U0. Dabei ist U0 die Leerlaufspannung der Quelle. Vielleicht haben Sie sich gewundert, daß der Klangregler eine Grunddämpfung von zirka 3 dB hatte, wo doch solche Schaltungen normalerweise für 0 dB Dämpfung dimensioniert sind. Hätte man die Spannung am Knoten 1 als Eingangsspannung genommen, dann wäre das auch rausgekommen. Sie können das leicht nachvollziehen, indem Sie den Quellwiderstand R0 ganz klein machen (zum Beispiel 1 Ohm). Dann ist die Spannung am Knoten 1 nämlich gleich

+ UE UA −

$$UA = V*UE$$

$$V = \frac{V0}{1 + J\frac{W}{W0}} \qquad W0 = \frac{2*\pi*FT}{V0}$$

Bild 3. Ersatzschaltbild eines Operationsverstärkers

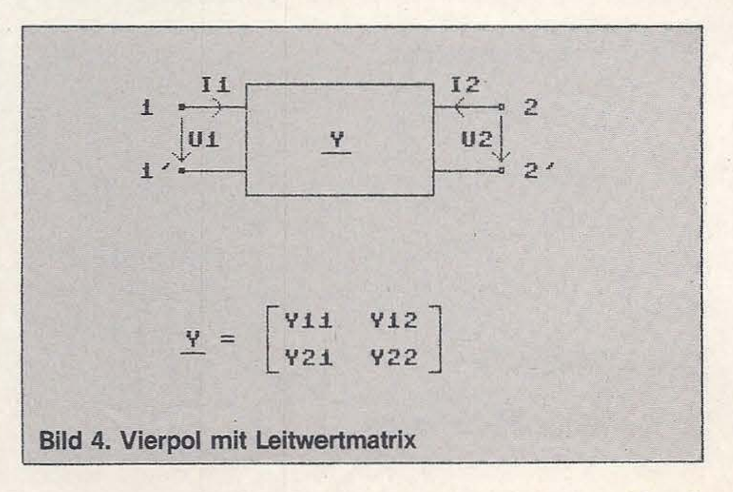


Bild 4. Vierpol mit Leitwertmatrix

UO. Das entspricht jedoch nicht der Praxis, denn auch dort hat die Quelle einen gewissen Innenwiderstand (Ausgangswiderstand der vorgeschalteten Stufe). Um diesen Sachverhalt zu berücksichtigen, verwendet die Definition des Übertragungsmaßes die Leerlaufspannung der Quelle.

Nun zur Ausgangsspannung: Das ist die Spannung an dem Knoten, den wir vorher als Ausgangsknoten angegeben haben, gemessen gegen Masse. In einer Brückenschaltung zum Beispiel interessiert uns aber die Brückenspannung UD, und die liegt nicht gegen Masse. Die Lösung dieses Problems ist jedoch ganz einfach: Wir nehmen einen Operationsverstärker mit der Leerlaufverstärkung VO = 1 und einer Transitfrequenz FT, die deutlich über der höchsten Meßfrequenz liegt. An den Eingang dieses Operationsverstärkers legen wir die Brückenspannung und haben sie dann am Ausgang gegen Masse.

Nun wissen wir, wie U0 und UA definiert sind und können daraus das Verhältnis UA/U0 bilden. Dieses Verhältnis gibt man üblicherweise in dB (Dezibel) an, und zwar nach der Formel: Übertragungsmaß in dB = 20xlog(UA/UO) (dekadischer Logarithmus)

Beispiele zu Vierpolen

Transformator, lose gekoppelt:

Y11 = L2/1
Y12 = Y21 = —M/1
Y22 = L1/1
dabei ist:
1 = L1xL2/(M^2)
L1 = Induktivität an 1 1'
L2 = Induktivität an 2 2'
M = Gegeninduktivität = $k \times \sqrt{L1 \times L2}$, wobei k der Kopplungsfaktor ist ($0<k<1$).

k darf den Wert 1 nicht erreichen (entspricht fester Kopplung), da sonst der Nenner 1 zu 0 würde. Ohm'sche Widerstände müssen in die Zuleitungen geschaltet werden, sie können nicht direkt in die Matrix aufgenommen werden, da der Vierpol keine inneren Knoten haben darf.

Idealer Gyrator

Y11 = Y22 = 0
Y12 = 1/r
Y21 = —1/r
r = Gyratorwiderstand

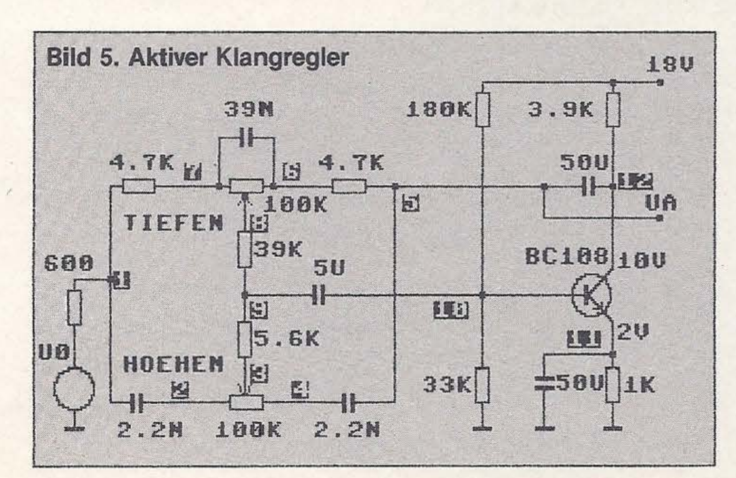
Durch Kettenschaltung zweiter Gyratoren kann ein idealer Übertrager (k = 1, L1, L2 gegen unendlich) nachgebildet werden, wobei ü = r1/r2.

Da jedoch keine galvanisch getrennten Schaltungsteile entstehen dürfen, müssen Sie beide Seiten eines Übertragers oder Gyrators irgendwo »anhängen«. Ferner können noch Transistoren, FETs, Röhren und eben alles, was sich in einer Y-Matrix, wie unter »Vierpol« beschrieben, darstellen läßt, mit Vierpolen nachgebildet werden.

Beschränkungen des Programms

Eine Schaltung darf maximal 20 Knoten und 90 Bauelemente beinhalten, wobei Transistoren, Operationsverstärker und Vierpole doppelt zählen. Außerdem ist die Zahl der Vierpole auf vier begrenzt. (Hans Haberl/ah)

Bild 5. Aktiver Klangregler



WIDERSTÄNDE R	1000	KAPAZITÄTEN C	3.3E-08
R 0 (0 1)	15000	C 1 (2 4)	3.3E-08
R 1 (1 2)	100	C 2 (2 3)	
R 2 (2 0)	120000	OPs + — A	V0,FT
R 3 (3 4)		OP 1 (3 0 4)	
		100000 000000	
		AUSGANGSKNOTEN	4

```
10 REM*********************************
20 REM*                               *
30 REM*            NEWEA2             *
40 REM*                               *
50 REM*       VON  HANS HABERL        *
60 REM*                               *
70 REM*********************************
100 REM  STEUERZEICHEN:
101 REM   "[CLR]" = SHIFT + CLR/HOME
102 REM   "[DOWN]" = CRSR DOWN
103 REM   "[UP]" = CRSR UP
104 REM   "[LEFT]" = CRSR BACK
105 REM   "[RVS ON]" = CTRL + 9
106 REM   "[RVS OFF]" = CTRL + 0
107 REM   "[CTRL+4]" = CTRL + 4
108 REM   "[CTRL+8]" = CTRL + 8
109 REM   "[C=3]" = COMMODORE + 3
110 REM   "[C=8]" = COMMODORE + 8
130 COLOUR6,6:PRINT"[CLR][C=8]":FILL1,0,40,23,81,7:FILL0,1
,38,25,81,7:FCHR3,3,34,19,32
150 PRINTAT(11,9)"AC-NETZWERKANALYSE":PRINTAT(12,1
5)"VON  HANS HABERL"
160 KEY1,"GOTO300"+CHR$(13)
170 DIMI,J,R,S,F,U,V,X,Y,P,Q,K,N,M,O,C,L,J1,J2,KF
180 DIMX1,X2,Y1,Y2,XF,XS,YF,YS,TL,TP,T0,Z,D,E,W,C$
,X$,N$,L%,Q%,I%
190 DIMR(20,21),S(20,21),M%(20,21),K(20),Y(2,20,20
),F(9,20),U(9,20),W(9,20)
200 DIMI(90),J(90),B(90),A(90),X(2,3,5),N(10),P(11
)
210 DIMC$(9),D$(9),E$(9),K$(9),B$(9)
220 TL=10/LOG(10):TP=180/π:T0=2*π:FORI=0TO9:READC$
(I),D$(I),E$(I):NEXTI
230 DATAF,KNOTEN ,ANZAHL DER ,A,WIDERSTAENDE,R,D,K
APAZITAETEN,C
240 DATAO,INDUKTIVITAETEN,L,N,TRANSISTOREN,C B E,E
,Y-VIERPOLE,1 1' 2 2'
250 DATAL,OP'S,+ - A,S,FREQUENZ,FREQUENZFILE,0,0,R
,0,0,W
280 I=0:REPEAT:GETC$:I=I+1:UNTILC$<>""ORI=300:PRIN
T"[CLR]":GOSUB500
300 CLOSE2:CLOSE4:CLOSE15
400 LOOP:PRINT"[DOWN][DOWN]>";:GOSUB5010
410 FORC=0TO7:IFC$<>C$(C)THENNEXTC:GOSUB500:END LO
OP
420 PRINT:ONC+1GOSUB1000,1000,1500,2000,3000,3300,
4050,4000:END LOOP
500 PRINT"[DOWN]MENUE:":PRINT"N=NETZWERK EINGEBEN":PRIN
T"E=ELEMENTE EINGEBEN"
510 PRINT"F="D$(7)"GANG":PRINT"A=AUSGABE":PRINT"D=
DIAGRAMM"
520 PRINT"L=DATEN LADEN":PRINT"S=DATEN SPEICHERN":
RETURN
1000 C$="":GOSUB5650:GOSUB5700:IFQ=0THENP=0:RETURN
1010 I=Q:IFC<1ANDU<9THENPRINTD$(7)"EN VON ";:GOSUB
5700
1020 RCOMP:IFQ>0ANDF(Q,0)>0THENFORJ=0TOF(Q,0)+1:F(
I,J)=F(Q,J):NEXTJ
1030 RCOMP:IFC=0THENPRINT"OHNE AENDERUNG ?";:GOSUB
5010
1040 Q=I:L=0:INPUT"NOTIZ";K$(Q):IFC<1THENW(Q,0)=C:
IFC$<>"J"THENL=1
1080 GOSUB1400:IFP<3ORC<0THENRETURN
1100 C=1:LOOP:PRINT"FILE[RVS ON]#[RVS OFF] , [RVS ON]N[RVS OFF]ORMIERUNG ODER [RVS ON]D
[RVS OFF]RUCKER?";
1110 GOSUB5010:J=ASC(C$)-48:L=0
1120 IFC$="N"THENGOSUB1300:END LOOP
1130 IFC$="D"THENGOSUB7200:END LOOP
1140 EXIT IFJ<1ORJ>9:EXIT IFF(J,0)=0
1150 GOSUB1200:END LOOP:RETURN
1200 GOSUB5900:P=1:L=1:REPEAT:F=F(Q,P):IFF(J,L)<FT
HENF=F(J,L)
1220 C$=STR$(F):PRINTRIGHT$(C$,LEN(C$)-1):PRINT"[UP]"
;
1230 IFF=F(Q,P)THENPRINTTAB(7)"[CTRL+4]";:X=U(Q,P)
1240 RCOMP:GOSUB5950:PRINTTAB(23):X=W(Q,P):GOSUB59
50:PRINT:PRINT"[UP][C=8]";:P=P+1
1250 IFF=F(J,L)THENPRINTTAB(15):X=U(J,L)
```

Listing 1. Listing zum Programm Netzwerkanalyse Vor dem Eingeben müssen Sie Simons-Basic laden.

```
1260 RCOMP:GOSUB5950:PRINTTAB(31):X=W(J,L):GOSUB59
50:L=L+1
1270 PRINT:UNTILP>F(Q,0)ORL>F(J,0)
1280 PRINTTAB(10)"▮FILE"Q"▮FILE"J:RETURN
1300 INPUT"BEZUGSFREQUENZ";F:INPUT"DB-WERT";S:FORI
=1TOF(Q,0)
1310 IFF=F(Q,I)THENR=S-U(Q,I):FORI=1TOF(Q,0):U(Q,I
)=U(Q,I)+R:NEXTI
1320 RCOMP:GOSUB1400:RETURN
1330 NEXTI:RETURN
1400 GOSUB5900:FORP=1TO20
1410 IFF(Q,P)=0ANDC=0THENL=1
1420 IFL=1THENPRINTF(Q,P);:INPUTF(Q,P):PRINT"▮▮"
1430 IFF(Q,P)>0THENONC+2GOSUB6000,4400,4700:NEXTP
1440 F(Q,0)=P-1:RETURN
1500 GOSUB5750:Q=P:IFQ=0THENRETURN
1600 GOSUB1700:FORI=1TOQ:FORJ=1TOF(P(I),0):X1=X2:Y
1=Y2
1610 X2=LOG(F(P(I),J))*XF+XS:Y2=U(P(I),J)*YF+YS
1620 IFJ>1THENLINEX1,Y1,X2,Y2,1:LINEX1+1,Y1,X2+1,Y
2,1
1630 NEXTJ:IFQ>1THENCHARX2-18,Y2-6,P(I)+48,1,1
1640 NEXTI:GOSUB5050:NRM:PRINT"▮AUSGABE AUF DRUCKE
R ?";:GOSUB5010
1650 IFC$="J"THENGOSUB7300
1660 RETURN
1700 X1=F(P(1),1):X2=X1+1:Y2=U(P(1),1):Y1=Y2-1
1710 FORI=1TOQ:P=P(I):R=F(P,F(P,0)):S=F(P,1):IFR>X
2THENX2=R
1720 IFS<X1THENX1=S
1730 FORJ=1TOF(P,0):R=U(P,J):IFR>Y2THENY2=R:ELSE:I
FR<Y1THENY1=R
1740 NEXTJ,I:YF=185/(Y1-Y2):YS=192-Y1*YF:XF=288/LO
G(X2/X1):XS=32-LOG(X1)*XF
1750 HIRES6,4:S=(Y2-Y1)/10:R=10↑INT(LOG(S)/LOG(10)
+.2):S=R*INT(S/R+1)
1760 I=0:REPEAT:I=I+1:Y=S*INT(Y1/S+I)
1770 IFY<Y2THENV=Y*YF+YS:TEXT0,V-3,STR$(Y),1,1,8:L
INE32,V,319,V,1
1780 UNTILY>Y2
1800 IFX2/X1>1.1THENGOSUB1820:ELSE:GOSUB1860
1810 RETURN
1820 S=LOG(X2/X1)/10:V=X2-X1:X=X1
1830 REPEAT:R=10↑INT(LOG(X*V/(X+V))/LOG(10)):X=R*I
NT(X/R+1):GOSUB1900
1840 X=X*EXP(S):UNTILX>X2:RETURN
1860 S=(X2-X1)/6:R=10↑INT(LOG(S)/LOG(10)+.2):S=R*I
NT(S/R+1)
1870 I=0:REPEAT:I=I+1:X=S*INT(X1/S+I):GOSUB1900:UN
TILX>X2:RETURN
1900 U=X:LOOP:EXIT IFU<1000:U=U/1000:END LOOP:C$=R
IGHT$(STR$(U),5)
1910 U=LOG(X)*XF+XS:TEXTU-4-4*LEN(C$),193,C$,1,1,8
:LINEU,0,U,191,1:RETURN
2000 C=-1:GOSUB1000:GOSUB5000:IFC$="J"ORP<2THENRET
URN
2010 L%=P-1:C=6:GOSUB5660:IFU>0THENP(10)=P(1):ELSE
:P(10)=9-INT(Q/9)
2020 REPEAT:PRINT"OPTIMIERUNG VON MAX. 4 ELEMENTEN
"
2030 P=1:REPEAT:PRINTP".ELEMENT "B$(P);:INPUTB$(P)
:GOSUB6050:UNTILP>4
2040 GOSUB5000:UNTILC$<>"J":P(0)=9-P:IFP=9THENRETU
RN
2050 P(9)=Q:P=P(10):F(P,0)=L%:W(P,0)=1:K$(P)="IST"
:Z=1:Q%=10:PRINT"X=ABBRUCH"
2100 LOOP:L=2:Q=P(Q%):FORP=1TOL%:F(Q,P)=F(P(9),P):
GOSUB4400:EXIT IFR=0
2110 W(Q,P)=U(Q,P)-U(P(Q%-1),P):NEXTP
2130 FORJ=10TOQ%:U=-R:R=0:FORP=1TOL%:R=R+W(P(Q%),P
)*W(P(J),P)/W(P(9),P)↑2
2140 NEXTP,J:IFQ%=11THENS=(1+ATN(U/R/6)/3)↑2:GOSUB
3500:Q%=10:END LOOP:Z=0:RETURN
2150 PRINT:PRINT"ZYKLUS"Z"FEHLER:"R:FORP=1TOP(0):P
RINTB$(P(P)),B(P(P));
2160 IFLEN(B$(P))>4THENPRINTB(P(P+4)):ELSE:PRINT
2170 NEXTP:GETC$
2180 IFC$="X"ORC<1THENPRINT"ABBRUCH?";:GOSUB5010:E
XIT IFC$="J"ORC$="X"
2200 Q%=11:I%=MOD(Z,P(0))+1:Z=Z+1:S=1.1:D=B(P(I%))
:E=B(P(I%+4)):GOSUB3500
2210 END LOOP:Z=0:RETURN
3000 J=J1:FORI=1TO5:P(I)=N(I):NEXTI:REPEAT:PRINT"▮
"E$(0);D$(0)"(OHNE 0)",N;
3010 INPUTN:FORI=1TO6:PRINTE$(0);D$(I)TAB(30)N(I);
:INPUTN(I):N(I)=ABS(N(I))
3020 NEXTI:GOSUB5000:GOSUB5600
3030 UNTILJ1<91ANDN>0ANDN<21ANDN(6)<6ANDC$<>"J"
3040 M=1:J=J+J1:FORI=1TO5:P=M+P(I)*(1+INT(I/4)):M=
M+N(I)*(1+INT(I/4))
3050 IFP>MTHENFORK=MTOJ:GOSUB5620:NEXTK
3060 IFP<MTHENFORK=JTOMSTEP-1:GOSUB5620:NEXTK
3070 NEXTI:REPEAT:K=1:FORI=1TO6:IFN(I)>0THENPRINT"
▮"D$(I)" AN "D$(0)
3080 RCOMP:FORJ=1TON(I):ONIGOSUB5100,5100,5100,515
0,5200,5150:GOSUB5250:NEXTJ
3090 NEXTI:REPEAT:PRINT"▮AUSGANGS"D$(0),N(9);:INPU
TN(9)
3100 UNTILN(9)>0ANDN(9)<=N
3110 GOSUB5000:UNTILC$<>"J":PRINT
3300 REPEAT:K=0:J(0)=1:FORI=1TO6:IFN(I)>0ORI=1THEN
PRINT:PRINTD$(I)" "E$(I)
3310 RCOMP:FORJ=SGN(I-1)TON(I):ONIGOSUB5300,5300,5
300,5350,5450,5400
3320 RCOMP:K=K+1+INT(I/4):NEXTJ
3330 NEXTI:GOSUB5000:UNTILC$<>"J"
3500 IFZ>0THENB(P(I%))=D*S:B(P(I%+4))=E*S↑SGN(49-P
(I%+4))
3600 GOSUB6200:U=-1:M=N+1:FORI=1TON:FORJ=1TOM:M%(I
,J)=0:NEXTJ,I
3610 K=1:J2=1:FORL=0TO2:FORI=1TON:FORJ=1TON:Y(L,I,
J)=0:NEXTJ,I
3620 J2=J2+N(L+1):LOOP:EXIT IFK=J2:I=I(K):J=J(K):G
OSUB5800:K=K+1
3630 END LOOP:NEXTL:Y(0,1,1)=Y(0,1,1)+1/B(0):M%(1,
M)=U
3650 J2=J2+2*N(4):LOOP:EXIT IFK=J2:L=0:S=ABS(B(K))
/.026:R=S*5E-4:P=I(K+1):O=I(K)
3660 I=P:J=O:GOSUB5820:J=J(K):R=S:GOSUB5820:O=J:R=
S/A(K):GOSUB5820
3670 L=1:R=S/T0/B(K+1):GOSUB5820:I=I(K):P=I:R=R*5E
-4:GOSUB5820:K=K+2:END LOOP
3700 S=1:J2=J2+2*N(5):LOOP:EXIT IFK=J2:I=I(K):J=J(
K):P=I:O=J:KF=0:GOSUB5850
3710 I=I(K+1):J=J(K+1):KF=1:GOSUB5850:P=I:O=J:KF=3
:GOSUB5850
3720 I=I(K):J=J(K):KF=2:GOSUB5850:S=S+1:K=K+2:END
LOOP
3750 J2=J2+2*N(6):LOOP:EXIT IFK=J2:L=I(K+1):FORI=0
TO2:FORJ=1TON:Y(I,L,J)=0
3760 NEXTJ,I:Y(0,L,L)=1:M%(L,L)=U:M%(L,I(K))=U:M%(
L,J(K))=U
3770 A(K)=T0*B(K+1)/B(K):K=K+2:END LOOP:J2=J2-2*N(
6)
3780 FORI=1TON:M%(I,0)=I:NEXTI
3790 P=N(9):IFP<NTHENQ=N:GOSUB6300
3800 FORI=1TON-1:FORJ=I+1TON:M%(I,J)=M%(I,J)ORM%(J
,I):M%(J,I)=M%(I,J):NEXTJ,I
3810 IFN<3THEN3920
3820 FORP=1TON-2:F=N*N:FORK=PTON-1:V=0:FORI=PTON-1
3830 IFM%(I,K)THENFORJ=I+1TON:W=M%(K,J)ANDNOTM%(I,
J):V=V-W:NEXTJ
3840 NEXTI:V=2*V-M%(K,M):IFV=FTHENGOSUB6350:IFR>ST
HENS=R:Q=K
3850 IFV<FTHENF=V:Q=K:GOSUB6350:S=R
3860 NEXTK:IFQ>PTHENGOSUB6300
3870 IFF=0THEN3910
3880 FORI=P+1TON-1
3890 IFM%(I,P)THENFORJ=I+1TON:M%(I,J)=M%(I,J)ORM%(
P,J):M%(J,I)=M%(I,J):NEXTJ
3900 NEXTI
3910 NEXTP
3920 FORI=1TON:M%(I,M)=M%(I,M)ORM%(I-1,M):K(M%(I,0
))=I:NEXTI:GOSUB6250
3930 IFZ>0THENRETURN
3950 PRINT"AUSGABE AUF DRUCKER?";:GOSUB5010:IFC$="
J"THENGOSUB7000
3960 RETURN
4000 F=0:OPEN15,8,15:PRINT"SCHALTUNG SPEICHERN ?";
:GOSUB5010
4010 IFC$="J"ANDN*J1>0THENGOSUB4100
4020 PRINT:IFF=0THENPRINTE$(7)"S SPEICHERN ?";:GOS
UB5010:IFC$="J"THENGOSUB4300
4030 CLOSE2:CLOSE15:RETURN
4050 OPEN15,8,15:GOSUB5500
4060 IFF=0THENINPUT#2,C$:IFRIGHT$(C$,5)="NEWEA"THE
NONVAL(C$)GOSUB4110,4300
4070 CLOSE15:CLOSE2:RETURN
4100 GOSUB5500:IFF>0THENRETURN:ELSE:PRINT#2,"1NEWE
A":INPUT"NOTIZ";N$
4110 PRINT:IFC=7THENPRINT#2,CHR$(34)+N$+CHR$(34):E
LSE:INPUT#2,N$:PRINTN$
4120 N(0)=N:FORI=0TO9:IFC=7THENPRINT#2,N(I):ELSE:I
NPUT#2,N(I)
4130 NEXTI:N=N(0):GOSUB5600:FORI=0TOJ1
4140 IFC=7THENPRINT#2,I(I):PRINT#2,J(I):PRINT#2,B(
I):PRINT#2,A(I):ELSE:INPUT#2,I
(I),J(I),B(I),A(I)
```

Listing 1. Listing zum Programm Netzwerkanalyse

```
4150 NEXTI:IFN(5)=0THEN4190
4160 FORK=1TON(5):FORI=0TO2:FORJ=0TO3
4170 IFC=7THENPRINT#2,X(I,J,K):ELSE:INPUT#2,X(I,J,
K)
4180 NEXTJ:NEXTI:NEXTK
4190 GOSUB5550:CLOSE2:IFF=0ANDC=6THENGOSUB3300
4200 RETURN
4300 ONC-5GOSUB5660,5750:IFP=0ORC+U=16THENRETURN
4310 IFC=7THENGOSUB5500:IFF=0THENPRINT#2,"2NEWEA":
PRINT#2,P:ELSE:IFF>0THENRETURN
4330 IFC=6THENINPUT#2,P:IFP>UTHENPRINT"NUR"U"VON"P
:P=U
4340 FORI=1TOP:Q=P(I)
4345 IFC=6THENINPUT#2,K$(Q):PRINTQ;K$(Q):ELSE:PRIN
T#2,CHR$(34)+K$(Q)+CHR$(34)
4350 J=0:REPEAT:IFC=6THENINPUT#2,F(Q,J),U(Q,J),W(Q
,J)
4360 IFC=7THENPRINT#2,F(Q,J):PRINT#2,U(Q,J):PRINT#
2,W(Q,J)
4370 J=J+1:UNTILJ>F(Q,0):F(Q,J)=0:NEXTI:GOSUB5550:
RETURN
4400 GOSUB6200:S=T0*F(Q,P):IFS=0ORN=0THENGOSUB6400
:RETURN
4410 FORI=1TON:U=K(I):FORJ=1TON:V=K(J):S(U,V)=Y(1,
I,J)*S-Y(2,I,J)/S
4420 R(U,V)=Y(0,I,J):NEXTJ:S(I,M)=0:R(I,M)=0:NEXTI
:R(K(1),M)=1/B(0)
4450 K=J2:LOOP:EXIT IFK=J2+2*N(6):I=K(I(K+1)):IFI=
0THEN4490
4460 R=1+S*S/A(K)↑2:U=B(K)/R:V=U*S/A(K)
4470 J=K(I(K)):R(I,J)=R(I,J)-U:S(I,J)=S(I,J)+V
4480 J=K(J(K)):R(I,J)=R(I,J)+U:S(I,J)=S(I,J)-V
4490 K=K+2:END LOOP
4500 FORK=1TON-1:U=R(K,K):V=S(K,K):F=U*U+V*V
4510 IFF=0THENGOSUB6400:RETURN
4550 FORI=K+1TON:IFNOTM%(I,K)THEN4600
4560 X=R(I,K):Y=S(I,K):R=(X*U+Y*V)/F:S=(Y*U-X*V)/F
:FORJ=K+1TOM
4580 IFM%(I,J)THEN:X=R(K,J):Y=S(K,J):R(I,J)=R(I,J)
-R*X+S*Y:S(I,J)=S(I,J)-S*X-R*Y
4590 NEXTJ
4600 NEXTI,K:U=R(N,N):V=S(N,N):F=U*U+V*V:IFF=0THEN
GOSUB6400:RETURN
4610 X=R(N,M):Y=S(N,M):R=(X*X+Y*Y)/F
4620 GOSUB6250:IFR=0THENPRINT"UA=0":P=P-1:L=1:RETU
RN
4630 U(Q,P)=TL*LOG(R):IFL=2THENRETURN
4640 R=X*U+Y*V:S=Y*U-X*V
4650 IFS=0THENW(Q,P)=(1-SGN(R))*90:ELSE:W(Q,P)=90*
SGN(S)-TP*ATN(R/S)
4700 X=F(Q,P):PRINTX;:PRINTSPC(12-LEN(STR$(X)));:X
=U(Q,P):GOSUB5950
4710 X=W(Q,P):PRINTSPC(7):GOSUB5950
4720 PRINT:IFL=0ORP<2THENRETURN
4750 F=F(Q,P):U=U(Q,P):V=W(Q,P):FORJ=PTO2STEP-1
4760 IFF(Q,J-1)>FTHENF(Q,J)=F(Q,J-1):U(Q,J)=U(Q,J-
1):W(Q,J)=W(Q,J-1):NEXTJ
4770 F(Q,J)=F:U(Q,J)=U:W(Q,J)=V:RETURN
5000 PRINT"▩KORREKTUR? ";
5010 FLASH7,20:PRINT"▥ ▓";:GOSUB5050:PRINT"▮ ▮"C$:
OFF:RETURN
5050 POKE198,0:WAIT198,1:GETC$:RETURN
5100 PRINT"  "E$(I);J,I(K);J(K);:INPUTI(K),J(K):RET
URN
5150 PRINT"  "LEFT$(D$(I),I-3);J,E$(I);I(K);J(K);I(
K+1);
5160 INPUTI(K),J(K),I(K+1):J(K+1)=0:RETURN
5200 REPEAT:PRINT"  VP"J,E$(I):PRINTI(K);J(K);I(K+1
);J(K+1);
5210 INPUTI(K),J(K),I(K+1),J(K+1):UNTILJ(K+1)<=N:R
ETURN
5250 FORQ=KTOK+I/4
5260 IFI(Q)<0ORJ(Q)<0ORN<I(Q)ORN<J(Q)THENJ=J-1:RET
URN
5270 NEXTQ:K=Q:RETURN
5300 REPEAT:GOSUB6500:INPUTB(K):UNTILB(K)<>0:RETUR
N
5350 REPEAT:GOSUB6550:INPUTA(K),B(K),B(K+1):UNTILB
(K+1)>0ANDA(K)>0:RETURN
5400 REPEAT:GOSUB6600:INPUTB(K),B(K+1):UNTILB(K+1)
>0:RETURN
5450 GOSUB6650:FORL=0TO3:GOSUB6660
5460 INPUTX(0,L,J),X(1,L,J),X(2,L,J):NEXTL:RETURN
5500 REPEAT:PRINT:INPUT"FILENAME ($=DIRECTORY)";N$
:IFN$="$"THENPRINT:DIR"$:*=S"
5510 PRINT:GOSUB5550:IFF>0THENRETURN
5520 UNTILN$<>"$":OPEN2,8,2,N$+",S,"+E$(C+2)
5550 PAUSE1:INPUT#15,F,C$,R,S:IFF>0THENPRINTF;C$;R
;S
5560 RETURN
5600 J1=N(1)+N(2)+N(3)+2*(N(4)+N(5)+N(6)):RETURN
5620 Q=K+P-M:I(K)=I(Q):J(K)=J(Q):B(K)=B(Q):A(K)=A(
Q):RETURN
5650 PRINT"▩▥":PRINTE$(7)"BELEGUNG:"
5660 U=0:FORI=1TO9:K=F(I,0):IFK=0THENU=U+1:P(U)=I
5670 IFK>0ANDC<>6THENPRINTI":"K$(I):PRINT"  "K"FREQ
:"F(I,1)"-"F(I,K)
5680 NEXTI:PRINT:P=U:RETURN
5700 REPEAT:REPEAT:PRINTE$(7)"  (1-9)"Q;:INPUTQ
5710 UNTILQ>=0ANDQ<=9:UNTILF(Q,0)>0ORC<10RQ=0:RETU
RN
5750 GOSUB5650:REPEAT:FORI=1TO9-U:PRINTI".";:Q=P(I
):GOSUB5700:P(I)=Q
5760 IFQ>0THENNEXTI
5770 P=I-1:GOSUB5000:UNTILC$<>"J":RETURN
5800 P=I:O=J:IFL=1THENR=B(K):ELSE:R=1/B(K)
5820 IFI=JORP=OTHENRETURN
5830 Y(L,P,I)=Y(L,P,I)+R:Y(L,O,J)=Y(L,O,J)+R:M%(P,
I)=U:M%(O,J)=U
5840 Y(L,P,J)=Y(L,P,J)-R:Y(L,O,I)=Y(L,O,I)-R:M%(P,
J)=U:M%(O,I)=U:RETURN
5850 FORL=0TO2:R=X(L,KF,S):IFR<>0THENGOSUB5820
5860 NEXTL:RETURN
5900 PRINT:PRINTD$(7);SPC(5)"BETRAG (DB)    ";
5910 IFW(Q,0)=0THENPRINT"PHASE (GRAD)":ELSE:PRINT"
TOLERANZ(DB)"
5920 RETURN
5950 X$=STR$(SGN(X)*INT(ABS(X)*1E3+.5))
5960 LOOP:EXIT IFLEN(X$)>4:X$=INSERT("0",X$,1):END
 LOOP
5970 X$=INSERT(".",X$,LEN(X$)-3):USE"####.###",X$:
RETURN
6000 PRINTDUP(" ",10)"▢":PRINTF(Q,P)TAB(8):X=U(Q,P
):GOSUB5950:INPUTU(Q,P)
6010 REPEAT:PRINT"▢"TAB(23):X=W(Q,P):GOSUB5950
6020 INPUTW(Q,P):UNTILW(Q,P)>0:GOSUB4750:RETURN
6050 R=1:C$=B$(P):L=LEN(C$):IFC$="0"THENP=10-P:RET
URN
6060 FORI=1TO3:FORJ=RTOR+2:IFMID$(C$,J,1)<>E$(I)TH
ENNEXTJ,I:RETURN
6070 K=VAL(RIGHT$(C$,L-J)):IFK<INT(I/2)ORK>N(I)THE
NRETURN
6080 FORJ=1TOI:K=K+N(J):NEXTJ:K=K-N(I):P(P+4)=K:IF
R=1THEN6120
6100 FORI=3TO4:N$=MID$(C$,I,1)
6110 IFN$="*"THENP(P+4)=K+50:ELSE:IFN$<>"/"THENNEX
TI:RETURN
6120 IFR=1THENP(P)=K:R=4:IFL>4THEN6060:ELSE:P=P+1:
RETURN
6200 PRINT"▩▩***  BUSY  ***▓":RETURN
6250 PRINT"▢"DUP(" ",14)"▢":RETURN
6300 FORI=1TON:R=M%(I,P):M%(I,P)=M%(I,Q):M%(I,Q)=R
:NEXTI
6310 FORJ=0TOM:R=M%(P,J):M%(P,J)=M%(Q,J):M%(Q,J)=R
:NEXTJ:RETURN
6350 R=U:FORJ=1TON:R=R-M%(K,J):NEXTJ:RETURN
6400 GOSUB6250:R=0:PRINT"NICHT LOESBAR":P=P-1:L=1:
RETURN
6500 PRINT"  "E$(I);J"("I(K);J(K)")",B(K);:RETURN
6550 PRINT"  T"J"("I(K);J(K);I(K+1)")  B0,IC,FT":PRI
NTA(K);B(K);B(K+1);:RETURN
6600 PRINT"  OP"J"("I(K);J(K);I(K+1)")  V0,FT":PRINT
B(K);B(K+1);:RETURN
6650 PRINT"  VP"J"("I(K);J(K);I(K+1);J(K+1)")  1/R
C  1/L":RETURN
6660 PRINT"  Y"11+8*INT(L/2)+L;X(0,L,J);X(1,L,J);X
(2,L,J);:RETURN
6700 GOSUB6650:FORL=0TO3:GOSUB6660:PRINT:NEXTL:RET
URN
7000 INPUT"NOTIZ";N$:GOSUB7400:PRINT"    "N$
7020 K=0:FORI=1TO6:IFN(I)>0ORI=1THENPRINT:PRINTD$(
I)"  "E$(I)
7030 RCOMP:FORJ=SGN(I-1)TON(I):ONIGOSUB6500,6500,6
500,6550,6700,6600
7040 RCOMP:PRINT:K=K+1+INT(I/4):NEXTJ
7050 NEXTI:PRINT:PRINT"AUSGANGSKNOTEN:",N(9):PRINT
#4:CLOSE4:RETURN
7100 GOSUB7400:PRINT"OPTIMIERUNGSVERLAUF":PRINT#4:
CLOSE4:RETURN
7200 GOSUB7400:PRINT"    "K$(Q):GOSUB1400:PRINT#4:C
LOSE4:RETURN
7300 GOSUB7400:FORI=1TOQ:IFQ>1THENPRINT"    "P(I);
7310 PRINT"    "K$(P(I)):NEXTI:PRINT:PRINT#4:CLOSE4
:COPY:RETURN
7400 OPEN4,4:CMD4:PRINT:PRINTCHR$(14)"AC-NETZWERKA
NALYSE"CHR$(15)
7410 PRINT:RETURN
```

Listing 1. Listing zum Programm Netzwerkanalyse (Schluß)

Tiny-Forth-Compiler zum Abtippen

Forth ist sicher eine der interessantesten Programmiersprachen überhaupt. Unser neues Listing des Monats stellt diese Sprache jedem C 64-Besitzer zur Verfügung.

Die folgende Anleitung zur Handhabung des Forth-Compilers kann natürlich kein Lehrbuch ersetzen. Falls Sie mit Forth noch keinerlei Erfahrungen haben, finden Sie im Anhang eine Übersicht über Forth-Literatur. Doch nun zur Beschreibung unseres Tiny-Forth-Compilers.

Die folgenden Befehle haben so gut wie alle einen Einfluß auf den Stack. Deshalb wird eine verkürzte Schreibweise verwendet, um das Verhalten der einzelnen Befehle darzustellen: »(Stack vorher - - - Stack nachher) «.

Die verwendeten Symbole haben folgende Bedeutung:
n = 16-Bit-Zahl, b = 8-Bit-Zahl, c = ASCII, addr = Adresse, f = Flag (0/1).

Die einzelnen Befehle des Tiny-Forth-Compilers sind in Tabelle 1 noch einmal übersichtlich dargestellt. Im folgenden werden die Befehle genauer beschrieben:

Arithmetikbefehle

Addition »+« (n2 n1 - - - n3)
Der Additionsbefehl holt die ersten beiden Argumente (n1,n2) vom Stack, addiert sie miteinander und legt das Ergebnis (n3) in den TOS (Top of Stack).

Subtraktion »-« (n2 n1 - - - n3)
Der Subtraktionsbefehl holt, wie bei der Addition, die ersten zwei Argumente (n1,n2) vom Stack, das Ergebnis (n3) wird wieder im TOS abgelegt.

Multiplikation »*« (n2 n1 - - - n3)
Die Multiplikation verhält sich analog zur Addition und Subtraktion.

Division »/« (n2 n1 - - - n3)
Analog zu Multiplikation.

Modulo »MOD« (n2 n1 - - - n3)
Ähnlich einer Division, es wird aber nicht das Ergebnis der Division, sondern der Divisionsrest auf den Stack gelegt.

Vergleichsbefehle

Gleich »=« (n2 n1 - - - f1)
Es werden die zwei obersten Werte auf Gleichheit geprüft.

Größer »<« (n2 n1 - - - f1)
Es wird geprüft, ob n2 größer als n1 ist.

Kleiner »>« (n2 n1 - - - f1)
Es wird geprüft, ob n2 kleiner als n1 ist.

Logische Verknüpfungen

AND (n2 n1 - - - n3)
Zwischen den Werten n1 und n2 wird eine logische UND-Operation ausgeführt.

OR = (n2 n1 - - - n3)
Es wird ein logisches ODER ausgeführt.

XOR (n2 n1 - - - n3)
Es wird ein logisches exklusives ODER ausgeführt.

NOT (f1 - - - f2)
Das oben liegende Flag wird invertiert.

Stackoperatoren

DROP (n2 n1 - - - n2)
Der oberste Wert wird vom Stack entfernt.

DUP (n1 - - - n1 n1)
Der oberste Wert auf dem Stack wird dupliziert.

SWAP (n2 n1 - - - n1 n2)
Die obersten beiden Werte werden vertauscht.

OVER (n2 n1 - - - n2 n1 n2)
Kopiert den zweiten Wert zum neuen TOS.

PICK (n1 n - - - n1 n2)
Pick holt den n-ten Wert in den TOS.

ROT (n3 n2 n1 - - - n2 n1 n3)
Rot läßt die ersten drei Elemente des Stack gegen den Uhrzeigersinn rotieren.

@ (addr - - - n1)
Holt eine 16-Bit-Zahl aus der Adresse addr.

! (n1 addr - - -)
Speichert eine 16-Bit-Zahl in der Adresse addr.

c@ (addr - - - b1)
Holt eine 8-Bit-Zahl aus der Adresse addr.

c! (b1 addr - - -)
Speichert eine 8-Bit-Zahl in der Adresse addr.

Kontrollstrukturen

BEGIN — UNTIL (f - - -)
Der Programmteil zwischen BEGIN und UNTIl wird solange ausgeführt, bis der TOS bei UNTIL ungleich Null ist.

BEGIN — WHILE — REPEAT (f - - -)
Der Programmteil zwischen BEGIN und REPEAT wird solange ausgeführt, wie der TOS bei WHILE ungleich Null ist.

IF — ENDIF (f - - -)
Der Programmteil zwischen IF und ENDIF wird nur dann ausgeführt, wenn der TOS bei IF ungleich Null ist.

IF — ELSE —ENDIF (f - - -)
Bei erfüllter Bedingung wird der Programmteil zwischen IF und ELSE ausgeführt, bei nichterfüllter Bedingung der zwischen ELSE und ENDIF.

Schleifen

DO (n2 n1 - - -)
DO legt die Argumente n1 (Endwert), n2 (Startwert) auf den Returnstack und leitet eine Schleife ein.

LOOP (- - -)
LOOP erhöht den Schleifen-Zähler um 1, ist der Endwert nicht erreicht, wird wieder zu dem auf DO folgenden Befehl gesprungen.

I (- - - n)
Der Befehl I legt den Wert des Schleifenzählers auf den TOS.

+LOOP (n - - -)
+LOOP erhöht den Schleifenzähler um n, weiter wie LOOP.

Ein-/Ausgabeoperatoren

KEY (- - - c)
Holt den ASCII-Wert, der gerade gedrückten Taste in den TOS, ist keine Taste gedrückt, so wird eine 0 in den TOS gelegt.

GET (- - - c)
Wartet, bis eine Taste gedrückt wird und legt dann ihren ASCII-Wert in den TOS.

EXPECT (addr n - - -)
Erwartet eine Eingabe, die mit Return abgeschlossen wird, und legt sie bei addr mit einer maximalen Länge n im Speicher ab. Als Abschlußzeichen wird eine 13 in den Speicher gesetzt.

EMIT (c - - -)
EMIT gibt den auf dem TOS liegenden ASCII-Wert auf dem Bildschirm aus.

TYPE (addr n - - -)
TYPE gibt n Zeichen, welche ab addr im Speicher stehen, auf dem Bildschirm aus.

CR (- - -)
CR bewirkt einen Zeilenvorschub.

CLS (- - -)
Löscht den Bildschirm.
."TEXT" (- - -)
Gibt den Text zwischen ».″« und »″« aus (funktioniert nur in compilierter Form).

Definitionsbefehle

n CONSTANT ⟨Name⟩ (n - - -)
CONSTANT definiert eine Konstante mit dem Wert n und dem Namen ⟨Name⟩. Wird ⟨Name⟩ im Programm aufgerufen, so wird n auf den TOS gelegt.
n VARIABLE ⟨Name⟩ (n - - -)
VARIABLE definiert eine Variable mit dem Wert n und dem Namen ⟨Name⟩. Wird ⟨Name⟩ im Programm aufgerufen, so wird die Adresse der Varible auf den Stack gelegt. Ein Wert wird mit @ (lies: Fetch) auf den TOS geholt und mit ! (lies: Store) an eine Variable übergeben.
n MEMORY ⟨Name⟩ (n - - -)
MEMORY definiert einen Speicherbereich mit dem Namen ⟨Name⟩ und der Länge n. Wird ⟨Name⟩ im Programm aufgerufen, so wird die Adresse des Speicherbereichs übergeben. In Adresse-2 ist die Länge zu finden und kann mit @ ausgelesen werden. MEMORY ist kein Standardwort!
: ⟨Name⟩...; (- - -)
Der Doppelpunkt definiert ein neues Forth-Wort mit dem Namen ⟨Name⟩. Die Definition muß mit »;« abgeschlossen werden.

Systembefehle

BASIC: Kehrt zum Basic zurück.
RESET: Kehrt zum Ausgangszustand zurück.
LIST n: Listet Screen n auf dem Drucker oder dem Bildschirm.
CLEAR n: Löscht SCREEN n auf der Diskette.
LOAD n: Compiliert Screen n in das Vocabulary.
FORGET ⟨Name⟩: Löscht das Wort ⟨Name⟩ aus dem Vocabulary.
VLIST: Listet das Vocabulary.
SAVE-SYSTEM ⟨Name⟩: Speichert den Objektcode und Vocabulary-Einträge aller selbstdefinierten Befehle auf Diskette.
LOAD-SYSTEM ⟨Name⟩: Lädt den Objektcode und die Vocabulary-Einträge wieder. Das System muß vorher mit RESET wieder in den Ausgangszustand gebracht werden und in Zeile 380 muß die Variable VOC denselben Wert wie beim Speichern haben, da sonst die Sprungadressen im Objektcode nicht stimmen.
EDIT n: Ruft den Bildschirmeditor auf, n ist die Nummer des Screens.

Spezialbefehle

CALL (addr - - -)
Ruft ein Maschinenprogramm mit der Adresse addr auf. Es können, wie bei SYS in Basic, in den Speicherzellen 780 das A-, 781 das X-, 782 das Y-Register mit übergeben werden.
R⟩ (- - - n)
Bringt das oberste Element des Return-Stacks auf den Stack.
⟩ R (n - - -)
Bringt das oberste Element des Stacks auf den Return-Stack.
;S (- - -)
Dieser Befehl sorgt für den vorzeitigen Abbruch des aktuellen Befehls.

Das Programm

Das Programm wurde mit Absicht sehr flexibel gehalten. Nach oder während des Abtippens können Sie einige Dinge nach Ihrem eigenen Ermessen ändern. So zum Beispiel die Startadresse des Objektcodes in Zeile 380; die Variable VOC enthält den Startwert. Allerdings sollten Sie nicht unter VOC=5*4096 gehen, da sonst eine Kollision mit dem String-Bereich möglich ist. Dies macht sich durch eine »OUT OF MEMORY«-Meldung oder durch einen Systemabsturz bemerkbar. Das Programm kann nur mit 16-Bit-Zahlen arbeiten und umfaßt nicht den gesamten Forth-Standard. Das Programm besteht aus einem Compiler, der in den Zeilen 1540 bis 2610, und einem Interpreter, der in den Zeilen 700 bis 1530 steht. Der Interpreter ist nicht in der Lage, alle Befehle auszuführen. Um Ihnen zu helfen, welche Befehle interpretierbar sind und welche nicht, sind nur die interpretierbaren im Vocabulary aufgeführt. Andere Befehle wie VARIABLE, MEMORY, etc., kurzum alle Definitionsbefehle, dürfen nicht im Compilermodus angewendet werden.

Das Programmieren in Forth

Nach dem Starten des Programms wird zuerst das Maschinenprogramm »VOCABULARY« in den Bereich 49152 bis 50160 geladen. Nach etwa einer Sekunde erscheint ein blinkender Cursor; jetzt können Sie Ihre Eingaben machen. Sie befinden sich im Interpretermodus, das heißt alle eingegebenen Befehle werden sofort ausgeführt. Daß dies im Interpretermodus so langsam geht, liegt nur daran, daß der Interpreter in Basic geschrieben ist; compilierte Befehle laufen dagegen etwa 10- bis 20mal so schnell wie Basic (sie werden vollkommen in Maschinensprache übersetzt). Vielleicht geben Sie mal das folgende Beispiel ein: »8 5 + .« (RETURN). Sie müßten jetzt 13 auf dem Bildschirm erhalten; wenn nicht, dann müssen Sie irgendwo im Programm einen Fehler gemacht haben. Sie können auch Kommentare einfügen, sie werden mit einer »(« begonnen und mit »)« abgeschlossen. Jeder Befehl wird durch ein Leerzeichen (oder Return) vom anderen getrennt.

Doch wie definiert man einen neuen Befehl? Diese Frage wird Ihnen sicher schon lange auf den Lippen brennen. Doch auch hier macht es Ihnen Forth sehr einfach. Um die Definition eines neuen Befehls einzuleiten wird »:« benutzt, gefolgt von dem Namen des neuen Befehls (bitte vergessen Sie nicht das Leerzeichen hinter jedem Befehl.) Dann folgen die Befehle, die in das Wort compiliert werden sollen. Ein »;« beendet die Definition. Danach ist der Befehl wie jeder andere Befehl benutzbar. Auf Fehler reagiert der Compiler, indem er die Compilation abbricht. Danach sollte man den Befehl mit FORGET löschen, da sonst das Programm abstürzt, wenn Sie den Befehl aufrufen.

Der Editor

Nun ist es ziemlich zeitaufwendig, wenn man bei jedem Fehler den Befehl neu eingeben muß, deshalb bietet Forth einen zweiten Editor. Es ist ein Bildschirmeditor, welcher mit dem Systembefehl »EDIT n« aufgerufen wird. n bezeichnet hier die Nummer des Screens, der editiert werden soll. Ein Screen ist einfach eine Bildschirmseite, auf der der zu compilierende Sourcecode steht.

Geben Sie einmal »EDIT 1« ein. Das System versucht nun, Screen 1 von der Diskette zu laden. Ist der Screen nicht auf Diskette vorhanden, so wird trotzdem in den Editor gesprungen, nur daß der Screen leer ist. Nachdem sich das System im Editor befindet, sehen Sie links die Zeilennummern von 0 bis 23 mit folgendem Doppelpunkt (dieser Doppelpunkt hat keine Bedeutung). Sie können nun mit dem Cursor hinter den Doppelpunkt fahren und eine Zeile eingeben. Jede Zeile muß mit RETURN abgeschlossen werden. Eine Zeile darf nicht länger als 35 Zeichen sein. Geben Sie doch einmal das vorherige Beispiel ein. Die Nummer am Anfang jeder Zeile entspricht in etwa einer Zeilennummer in Basic.

Um nun den Editor zu verlassen, gibt es zwei Möglichkeiten, einmal mit »e«, damit der Screen n abgespeichert wird und mit »s«, so wird der Editor ohne Änderung des Screens verlassen. Die Buchstaben müssen in der ersten Spalte einer Zeile stehen, also dort, wo die Zeilennummer steht. Es gibt noch mehr dieser Editorbefehle (siehe Tabelle 4). Zum Einfügen von Zeilen benützen Sie »I Zeile Anzahl«, mit »D Zeile Anzahl« löschen Sie Zeilen. Mit »L« listen Sie den Screen noch einmal, das ist dann von Nutzen, wenn Sie versehentlich die CLR-Taste betätigt haben. Mit »N Nummer« ändern Sie die Nummer eines Screens. Das Compilieren eines Screens geschieht mit LOAD n. Soll der nächste Screen (n+1) auch noch compiliert werden, so muß in der letzten Zeile des Screens n der Befehl »-- > « vorhanden sein. Experimentieren Sie doch mal ein bißchen mit dem Editor.

In den Bildern 2 bis 6 finden Sie einige selbstdefinierte Befehle. Die Befehle J und LEAVE in Bild 3 möchte ich näher erklären. J gibt den Schleifenwert der zweitinnersten Schleife aus. Die Befehle R> und >R manipulieren den Returnstack (der Returnstack funktioniert genauso wie der normale Stack). Hier werden Werte für Schleifen zwischengespeichert und zwar im Format Endwert, Zähler. R> holt den obersten Wert des Returnstacks auf den Stack, >R tut das Gegenteil. Die 704 ist nur Zwischenspeicher. LEAVE schließt eine Schleife vorzeitig ab, indem Endwert und Zähler gleichgesetzt werden.

Viel Spaß beim Programmieren in Forth.

(Alexander Schindowski/ev)

Info: Literatur zu Forth:
- Paul M. Chirlien, Der Einstieg in Forth, Markt & Technik 1985, 338 Seiten, 58 Mark.
- E. Floegel, Forth-Handbuch, Ing. W. Hofacker Verlag 1983, 192 Seiten, 39 Mark.
- Monadjemi, Das Trainingsbuch zu Forth, Data Becker 1985, 300 Seiten, 39 Mark.
- Ronald Zech, Die Programmiersprache Forth, Franzis-Verlag 1984, 312 Seiten, 69 Mark.
- Weitere Literatur mit Informationen über Forth sind zu beziehen über die deutsche Sektion der Forth Interest Group (FIG). Kontaktadresse: Forth Gesellschaft Deutschland (F16), Postfach 202264, 2000 Hamburg 20. Da die FIG nicht kommerziell arbeitet, bitte bei Anfragen Freiumschlag beifügen.

Befehl	Stack
1) Rechenoperationen:	
1.1) +	(n2 n1 - - - n3)
1.2) -	(n2 n1 - - - n3)
1.3) *	(n2 n1 - - - n3)
1.4) /	(n2 n1 - - - n3)
1.5) MOD	(n2 n1 - - - n3)
2) Vergleichsoperatoren:	
2.1) =	(n2 n1 - - - f)
2.2) >	(n2 n1 - - - f)
2.3) <	(n2 n1 - - - f)
3) Logische Verknüpfungen:	
3.1) AND	(n2 n1 - - - n3)
3.2) OR	(n2 n1 - - - n3)
3.3) NOT	(f1 - - - f2)
3.4) XOR	(n2 n1 - - - n3)
4) Stackoperatoren:	
4.0) DROP	(n1 - - -)
4.1) DUP	(n1 - - - n1 n1)
4.2) SWAP	(n2 n1 - - - n1 n2)
4.3) OVER	(n2 n1 - - - n2 n1 n2)
4.4) PICK	(n1 n - - - n1 n2)
4.5) ROT	(n3 n2 n1 - - - n2 n1 n3)
4.6) @	(addr - - - n1)
4.7) !	(n1 addr - - -)
4.8) c@	(addr - - - b1)
4.9) c!	(b1 addr - - -)
5) Kontrollstrukturen:	
5.1) BEGIN — UNTIL	(f - - -)
5.2) BEGIN — WHILE — REPRAT	(f - - -)
5.3) IF — ENDIF	(f - - -)
5.4) IF — ELSE — ENDIF	(f - - -)
6) Definitionsworte:	
6.1) VARIABLE	(n - - -)
6.2) CONSTANT	(n - - -)
6.3) MEMORY	(n - - -)
6.4) : ... ;	
7) Ein-/Ausgabe:	
7.1) EXPECT	(addr n - - -)
7.2) TYPE	(addr n - - -)
7.3) KEY	(- - - c)
7.4) GET	(- - - c)
7.5) EMIT	(c - - -)
7.6) ."	(- - -)
7.7) .	(n - - -)
7.8) CR	(- - -)
7.9) CLS	(- - -)
8) Schleifen:	
8.1) DO	(n2 n1 - - -)
8.2) LOOP	(- - -)
8.3) +LOOP	(n - - -)
8.4) I	(- - - n)
9) Sonstige Befehle:	
9.1) call	(addr - - -)
9.2) >R	(n - - -)
9.3) R>	(- - - n)
9.4) DEPTH	(- - - n)

Tabelle 1. Der Befehlssatz des Tiny-Forth-Compilers

A,X,I,Z	**Lauf- und Hilfsvariablen**
VOC	Vocabulary-Zeiger
BE	Beginn des Vocabulary
SP	Zeiger für Compilerstack
AN	Anzahl der Vocabulary-Einträge
ZE$	Eingabezeile
BE$	Einzelner Befehl
COMP	Flag für Compilation
AD	Adressen-Zwischenspeicher
OK	Zahlenumwandlung geglückt?
A$	Hilfsvariable
WO$(X)	Wörterverzeichnis
AD(X)	Startadressen-Verzeichnis
BLOCK	Flag für Compilation von Diskette
Z1	Zeilenzähler
XX	Umgewandelte Zahl
X	Top of Stack (TOS)

Tabelle 2. Variablenübersicht zum Tiny-Forth-Compiler

Befehl	Funktion
BASIC	Zurück zu Basic
RESET	Zurück in Ausgangsstellung
VLIST	Listet das Vocabulary
FORGET	Lösche ein Wort
FLOPPY	Gibt einen Befehl an die Floppy
SAVE-SYSTEM	Speichert das Vocabulary ab
LOAD-SYSTEM	Lädt das Vocabulary
LIST n	Listed eine Screen
LOAD n	Compiliert eine Screen
CLEAR n	Löscht eine Screen
EDIT n	Ruft den Editor auf

Tabelle 3. Forth-Systembefehle

Befehl	Funktion
E	Beendet das Editieren und speichert die Screens ab
S	Beendet das Editieren
L	Listet die Screens
I (z,a)	Fügt Zeile(n) ein
D (z,a)	Löscht Zeile(n)
N (x)	Ändert die Screen-Nummer

Tabelle 4. Befehle des Forth-Editors

Listing 1. Tiny-Forth-Compiler. Beachten Sie bitte die Eingabehinweise auf Seite 53.

```
340 IF A=0 THEN A=1:LOAD"VOCABULARY",8,1        <086>
350 DEF FN H(X)=(INT(X/256))                     <191>
360 DEF FN L(X)=(X-256*FN H(X))                  <030>
370 POKE 53272,23:PRINT"{CLR,LIG.BLUE}";CH
    R$(8);                                       <131>
380 VOC=6*4096:BE=VOC:SP=0:Z1=0                  <155>
390 POKE 55,FN L(BE):POKE 56,FN H(BE)            <236>
395 DIM ST(20),SC$(24),WO$(100),AD(100)          <114>
400 PRINT TAB(14);"FORTH-COMPILER"               <018>
410 PRINT TAB(17);"FUER DEN"                     <115>
420 PRINT TAB(15);"COMMODORE-64"                 <086>
430 PRINT"-----------------------------------
    ---------";                                  <156>
440 PRINT"{5SPACE}VON ALEXANDER SCHINDOWSK
    I 1985{8DOWN}"                               <092>
450 DATA 38                                      <238>
460 DATA "+",49563                               <028>
470 DATA "CLS",49158,"DEPTH",49968               <242>
480 DATA "@",50012,"DROP",49236                  <176>
490 DATA "EMIT",49855,"EXPECT",49936             <206>
500 DATA "=",49410,"I",49766                     <250>
510 DATA "KEY",49880                             <164>
520 DATA "+LOOP",49821,"MOD",49733               <091>
530 DATA "NOT",49458,"OVER",49284                <027>
540 DATA ".",49163,"-",49578                     <147>
550 DATA "SWAP",49248,">R",49751                 <128>
560 DATA "AND",49497,"CR",49384                  <086>
570 DATA "/",49721,"DO",49757,"!",49977          <084>
580 DATA "DUP",49239,"XOR",49541                 <015>
590 DATA "GET",49862,">",49434                   <131>
600 DATA "<",49452,"LOOP",49811                  <134>
610 DATA "*",49596,"OR",49519                    <164>
620 DATA "C@",50030,"C!",49996                   <225>
630 DATA "R>",49745,"TYPE",49915                 <188>
640 DATA "PICK",50062,"CALL",50047,"ROT",5
    0085                                         <056>
650 READ AN                                      <154>
660 FOR I=1 TO AN                                <080>
670 READ WO$(I),AD(I)                            <061>
680 NEXT I:POKE 2,0:POKE 252,0                   <134>
690 GOSUB 3830                                   <122>
693 :                                            <161>
695 REM **************************               <227>
700 REM *** BEFEHLS-AUSWERTUNG ***               <130>
705 REM **************************               <237>
708 :                                            <176>
710 GOSUB 2630                                   <102>
715 :                                            <183>
720 IF BE$=":" THEN 1540                         <051>
725 :                                            <193>
730 FOR I=AN TO 1 STEP -1                        <033>
740 IF BE$=WO$(I) THEN 760                       <118>
750 NEXT I:GOTO 770                              <178>
760 SYS AD(I):GOTO 700                           <092>
765 :                                            <233>
770 GOSUB 3030                                   <076>
780 IF OK=0 THEN 830                             <001>
790 POKE 781,FN L(XX)                            <010>
800 POKE 780,FN H(XX)                            <102>
810 SYS 49194                                    <108>
820 GOTO 700                                     <034>
825 :                                            <039>
830 IF BE$="RESET" THEN RUN                      <035>
835 :                                            <049>
840 IF BE$="BASIC" THEN END                      <031>
845 :                                            <059>
850 IF BE$<>"VLIST" THEN 900                     <240>
860 PRINT:FOR I=AN TO 1 STEP-1                   <174>
870 PRINT WO$(I)"{2SPACE}";                      <128>
880 NEXT:PRINT                                   <234>
890 GOTO 700                                     <104>
895 :                                            <109>
900 IF BE$<>"FORGET" THEN 950                    <147>
910 GOSUB 2630:FOR I=AN TO 1 STEP-1              <217>
920 IF BE$<>WO$(I) THEN NEXT I                   <108>
930 IF I>AN THEN PRINT BE$" I CAN'T FIND":
    GOTO 700                                     <050>
935 :                                            <149>
940 VOC=AD(I):AN=I-1:GOTO 700                    <163>
950 IF BE$<>"(" THEN 980                         <117>
960 IF BE$<>")" THEN GOSUB 2630:GOTO 960         <027>
970 GOTO 700                                     <184>
975 :                                            <189>
980 IF BE$<>"EDIT" THEN 1020                     <027>
990 GOSUB 2630 :SC=VAL(BE$)                      <196>
1000 PRINT"SCREEN: ";SC:GOSUB 3280               <163>
1010 IF BE$="-->"THEN ZE$="":SC=SC+1:GOTO
     1000                                        <005>
1012 GOTO 700                                    <226>
1015 :                                           <229>
1020 IF BE$<>"LOAD" THEN 1050                    <066>
1030 GOSUB 2630:SC=VAL(BE$)                      <238>
1040 BLOCK=1:Z1=0:GOSUB 3110:GOTO 700            <233>
1050 IF BE$<>"-->" THEN 1070                     <048>
1060 SC=SC+1:GOSUB 3110:COMP=1:BLOCK=1:Z1=
     0:GOTO 700                                  <225>
1070 :                                           <030>
1080 IF BE$<>"VARIABLE" THEN 1145                <137>
1085 GOSUB 2630:AN=AN+1:WO$(AN)=BE$              <243>
1090 AD(AN)=VOC:XX=VOC+10                        <131>
1095 GOSUB 3470:POKE VOC,169                     <027>
1100 POKE VOC+1,FN H(XX)                         <181>
1105 POKE VOC+2,162                              <127>
1110 POKE VOC+3,FN L(XX)                         <208>
1115 POKE VOC+4,32:POKE VOC+5,42                 <253>
1120 POKE VOC+6,192:POKE VOC+7,56                <091>
1125 POKE VOC+8,176:POKE VOC+9,2                 <139>
1130 POKE VOC+11,FN L(X)                         <120>
1135 POKE VOC+10,FB H(X)                         <011>
1140 GOTO 700                                    <100>
1145 :                                           <105>
1150 IF BE$<>"MEMORY" THEN 1220                  <167>
1155 GOSUB 2630:AN=AN+1:WO$(AN)=BE$              <057>
1160 AD(AN)=VOC                                  <061>
1165 GOSUB 3470:POKE VOC,169                     <097>
1170 POKE VOC+1,FN H(VOC+12)                     <205>
1175 POKE VOC+2,162                              <197>
1180 POKE VOC+3,FN L(VOC+12)                     <232>
1185 POKE VOC+4,32:POKE VOC+5,42                 <067>
1190 POKE VOC+6,192:AD=VOC+12+XX                 <244>
1195 POKE VOC+7,96                               <218>
1200 POKE VOC+8,FN L(AD):POKE VOC+9,FN H(A
     D)                                          <111>
1205 POKE VOC+10,FN L(XX):POKE VOC+11,FN H
     (XX)                                        <126>
1210 VOC=AD:GOTO 700                             <062>
1220 :                                           <180>
1230 IF BE$<>"CONSTANT" THEN 1280                <052>
1240 GOSUB 2630:A$=": "+BE$+" "                  <025>
1250 GOSUB 3470                                  <238>
1260 ZE$=A$+STR$(X)+" ;"+ZE$                     <110>
1270 GOTO 700                                    <230>
1280 :                                           <242>
1290 IF BE$<>"CLEAR" THEN 1350                   <152>
1300 GOSUB 2630:SC=VAL(BE$)                      <254>
1310 FOR ZE=0 TO 24                              <162>
1320 SC$(ZE)=""                                  <063>
1330 NEXT ZE:GOSUB 3220                          <005>
1340 GOTO 700                                    <046>
1350 :                                           <056>
1360 IF BE$="SAVE-SYSTEM"THEN 3510               <219>
1365 :                                           <071>
1370 IF BE$="LOAD-SYSTEM"THEN 3720               <159>
1380 :                                           <086>
1390 IF BE$<>"FLOPPY" THEN 1420                  <144>
1400 GOSUB 2630                                  <030>
1410 OPEN 1,8,15,BE$:CLOSE 1:GOTO 700            <181>
1420 :                                           <126>
1430 IF BE$<>"LIST" THEN 1520                    <088>
1440 GOSUB 2630:SC=VAL(BE$):GOSUB 3110           <018>
1450 INPUT"AUF DRUCKER (Y/N)";A$:A=3:IF A$
     ="Y"THEN A=4                                <097>
1460 OPEN 4,A,-7*(A=4)                           <220>
1470 FOR Z=0 TO 23                               <070>
1480 PRINT#4,RIGHT$(STR$(Z),2)":"SC$(Z)          <023>
1490 NEXT Z:CLOSE 4                              <124>
1500 IF A=3 THEN POKE 198,0:WAIT 198,1           <175>
1510 COMP=0:GOTO 700                             <198>
1520 :                                           <226>
1530 PRINT BE$" I CAN'T FIND":GOTO 700           <190>
1533 :                                           <239>
1535 REM **************************              <143>
1540 REM ***     COMPILER       ***              <216>
1545 REM **************************              <155>
1548 :                                           <000>
```

```
1550 GOSUB 2630:AN=AN+1:WO$(AN)=BE$             <200>
1560 AD(AN)=VOC:COMP=1                          <206>
1570 :                                          <022>
1580 GOSUB 2630                                 <212>
1590 FOR I=1 TO ANZ                             <174>
1600 IF BE$<>WO$(I) THEN NEXT I                 <026>
1610 AD=AD(I)                                   <144>
1615 :                                          <067>
1620 IF BE$<>"BEGIN" THEN 1640                  <215>
1630 ST(SP)=VOC:SP=SP+1:GOTO 1570               <147>
1635 :                                          <087>
1640 IF BE$<>"UNTIL" THEN 1730                  <091>
1650 POKE VOC,32                                <192>
1660 POKE VOC+1,180:POKE VOC+2,194              <254>
1670 POKE VOC+3,176:POKE VOC+4,3                <041>
1680 POKE VOC+5,76                              <190>
1690 SP=SP-1:AD=ST(SP):IF SP<0 THEN 65535       <250>
1700 POKE VOC+6,FN L(AD)                        <176>
1710 POKE VOC+7,FN H(AD)                        <042>
1720 VOC=VOC+8:GOTO 1570                        <124>
1725 :                                          <177>
1730 IF BE$=";" THEN POKE VOC,96:VOC=VOC+1
     :COMP=0:GOTO 700                           <150>
1735 :                                          <187>
1740 GOSUB 3030                                 <028>
1750 IF OK=0 THEN 1800                          <085>
1760 POKE VOC,169:POKE VOC+1,FN H(XX)           <082>
1770 POKE VOC+2,162:POKE VOC+3,FN L(XX)         <117>
1780 POKE VOC+4,32:POKE VOC+5,42                <154>
1790 POKE VOC+6,192:VOC=VOC+7:GOTO 1570         <174>
1800 :                                          <254>
1810 IF BE$<>"IF" THEN 1870                     <083>
1820 POKE VOC,32:POKE VOC+1,180                 <125>
1830 POKE VOC+2,194:POKE VOC+3,176              <146>
1840 POKE VOC+4,3:POKE VOC+5,76                 <049>
1850 ST(SP)=VOC+6:SP=SP+1                       <123>
1860 VOC=VOC+8:GOTO 1570                        <010>
1870 :                                          <068>
1880 IF BE$<>"ENDIF" THEN 1930                  <240>
1890 SP=SP-1:AD=ST(SP)                          <222>
1900 POKE AD,FN L(VOC)                          <247>
1910 POKE AD+1,FN H(VOC)                        <020>
1920 GOTO 1570                                  <130>
1930 :                                          <128>
1940 IF BE$<>"ELSE" THEN 2010                   <225>
1950 AD=ST(SP-1)                                <077>
1960 ST(SP-1)=VOC+1                             <023>
1970 POKE VOC,76:VOC=VOC+3                      <121>
1980 POKE AD,FN L(VOC)                          <071>
1990 POKE AD+1,FN H(VOC)                        <100>
2000 GOTO 1570                                  <210>
2010 :                                          <208>
2020 IF BE$="WHILE" THEN 1820                   <170>
2030 :                                          <228>
2040 IF BE$<>"REPEAT" THEN 2110                 <060>
2050 AD=ST(SP-1):A2=ST(SP-2)                    <184>
2060 SP=SP-1                                    <154>
2070 POKE VOC,76                                <110>
2080 POKE VOC+1,FN L(A2)                        <036>
2090 POKE VOC+2,FN H(A2)                        <230>
2100 VOC=VOC+3:GOTO 1980                        <106>
2110 :                                          <054>
2120 IF BE$<>"."+CHR$(34) THEN 2225             <144>
2125 A$="":ZE$=MID$(ZE$,2)                      <022>
2130 IF LEFT$(ZE$,1)<>CHR$(34) THEN A$=A$+
     LEFT$(ZE$,1):ZE$=MID$(ZE$,2):GOTO 213
     0                                          <185>
2135 ZE$=MID$(ZE$,2):A$=A$+CHR$(0)              <063>
2140 AD=VOC+10                                  <200>
2145 POKE VOC,169                               <040>
2150 POKE VOC+1,FN H(AD)                        <227>
2155 POKE VOC+2,162                             <161>
2160 POKE VOC+3,FN L(AD)                        <254>
2165 POKE VOC+4,32:POKE VOC+5,234               <165>
2170 POKE VOC+6,194:POKE VOC+7,76               <173>
2175 AD=VOC+10+LEN(A$)                          <012>
2180 POKE VOC+8,FN L(AD)                        <149>
2185 POKE VOC+9,FN H(AD)                        <010>
2190 VOC=VOC+10                                 <020>
2200 FOR I=0 TO LEN(A$)-1                       <150>
2205 POKE VOC+I,ASC(MID$(A$,I+1,1))             <036>
2210 IF LEFT$(ZE$,1)=" " THEN ZE$=MID$(ZE$
     ,2):GOTO 2210                              <173>
2215 NEXT I                                     <011>
2220 VOC=AD:GOTO 1570                           <190>
2225 :                                          <169>
2230 IF BE$<>"TEXT"+CHR$(34) THEN 2320          <213>
2235 A$="":ZE$=MID$(ZE$,2)                      <132>
2240 IF LEFT$(ZE$,1)<>CHR$(34) THEN A$=A$+
     LEFT$(ZE$,1):ZE$=MID$(ZE$,2):GOTO 224
     0                                          <045>
2245 ZE$=MID$(ZE$,2):A$=A$+CHR$(0)              <173>
2250 AD=VOC+10                                  <054>
2255 POKE VOC,169                               <150>
2260 POKE VOC+1,FN H(AD)                        <081>
2265 POKE VOC+2,162                             <015>
2270 POKE VOC+3,FN L(AD)                        <108>
2273 POKE VOC+4,32:POKE VOC+5,42:POKE VOC+
     6,192                                      <097>
2275 POKE VOC+7,76                              <022>
2280 AD=VOC+10+LEN(A$)                          <117>
2285 POKE VOC+8,FN L(AD)                        <254>
2290 POKE VOC+9,FN H(AD)                        <115>
2295 VOC=VOC+10                                 <125>
2300 FOR I=0 TO LEN(A$)-1                       <250>
2305 POKE VOC+I,ASC(MID$(A$,I+1,1)):NEXT        <041>
2310 IF LEFT$(ZE$,1)=" " THEN ZE$=MID$(ZE$
     ,2):GOTO 2310                              <083>
2315 VOC=AD:GOTO 1570                           <031>
2320 :                                          <010>
2330 IF BE$<>"DO" THEN 2390                     <088>
2340 POKE VOC,32                                <120>
2350 POKE VOC+1,FN L(AD)                        <189>
2360 POKE VOC+2,FN H(AD)                        <056>
2370 VOC=VOC+3:ST(SP)=VOC                       <014>
2380 SP=SP+1:GOTO 1570                          <196>
2390 :                                          <080>
2400 IF BE$<>"LOOP" AND BE$<>"+LOOP" THEN
     2500                                       <222>
2410 POKE VOC,32                                <190>
2420 POKE VOC+1,FN L(AD)                        <003>
2430 POKE VOC+2,FN H(AD)                        <126>
2440 POKE VOC+3,176:POKE VOC+4,3                <049>
2450 SP=SP-1:AD=ST(SP)                          <018>
2460 POKE VOC+5,76                              <208>
2470 POKE VOC+6,AD-256*INT(AD/256)              <142>
2480 POKE VOC+7,INT(AD/256)                     <091>
2490 VOC=VOC+8:GOTO 1570                        <132>
2500 :                                          <190>
2510 IF BE$<>"(" THEN 2540                      <225>
2520 GOSUB 2630:IF BE$<>")" THEN 2520           <128>
2530 GOTO 1570                                  <232>
2540 :                                          <230>
2550 IF BE$=";S" THEN POKE VOC,96:VOC=VOC+
     1:GOTO 1570                                <005>
2560 :                                          <252>
2570 IF I>AN THEN PRINT BE$" I CAN'T FIND"
     :COMP=0:GOTO 700                           <030>
2575 :                                          <011>
2580 POKE VOC,32                                <106>
2590 POKE VOC+1,AD-256*INT(AD/256)              <133>
2600 POKE VOC+2,INT(AD/256)                     <083>
2610 VOC=VOC+3:GOTO 1570                        <234>
2615 :                                          <051>
2620 REM ************************               <133>
2630 REM ** HOLE BEFEHL IN BE$ **               <160>
2635 REM ************************               <148>
2637 :                                          <073>
2640 IF ZE$="" THEN GOSUB 2750                  <086>
2650 IF LEFT$(ZE$,1)=" "THEN ZE$=MID$(ZE$,
     2):GOTO 2650                               <108>
2660 BE$="":FOR I=1 TO LEN(ZE$)                 <059>
2670 IF LEFT$(ZE$,1)=" " THEN 2710              <174>
2680 BE$=BE$+LEFT$(ZE$,1)                       <085>
2690 ZE$=MID$(ZE$,2)                            <164>
2700 NEXT I                                     <244>
2710 RETURN                                     <228>
2720 :                                          <156>
2730 REM *************************              <068>
2740 REM *** HOLE ZEILE IN ZE$ ***              <027>
2750 REM *************************              <088>
2755 :                                          <191>
2760 IF BLOCK=1 THEN 2880                       <023>
2770 IF COMP=0 THEN PRINT"{2SPACE}OK."          <039>
2780 SYS 42336                                  <123>
2790 ZE$=""                                     <190>
2800 FOR Z=512 TO 600                           <099>
2810 A=PEEK(Z)                                  <003>
```

Listing 1. Tiny-Forth-Compiler (Fortsetzung)

```
2820 IF A=0 THEN 2850                          <231>
2830 ZE$=ZE$+CHR$(A)                           <153>
2840 NEXT Z                                    <010>
2850 IF LEFT$(ZE$,1)=" "THEN ZE$=MID$(ZE$,
     2):GOTO 2850                              <183>
2860 IF ZE$="" THEN 2770                       <066>
2870 RETURN                                    <134>
2880 ZE$=SC$(Z1):PRINT RIGHT$(STR$(Z1),2);
     ":";ZE$                                   <219>
2890 IF LEN(ZE$)<2 THEN ZE$="({2SPACE})"       <071>
2900 Z1=Z1+1                                   <164>
2910 IF Z1=24 THEN BLOCK=0                     <186>
2920 RETURN                                    <184>
2980 :                                         <162>
2990 REM *************************             <236>
3000 REM **    WANDELE ZAHL UM     **          <118>
3010 REM **          IN XX         **          <069>
3020 REM *************************             <010>
3030 :                                         <212>
3040 OK=1:X=1                                  <172>
3050 IF LEFT$(BE$,1)="-" AND VAL(BE$)<0 TH
     EN BE$=MID$(BE$,2):X=-1:GOTO 3080         <007>
3060 IF LEFT$(BE$,1)>="0" AND LEFT$(BE$,1)
     <="9" THEN 3080                           <135>
3070 OK=0:RETURN                               <177>
3080 XX=VAL(BE$)*X                             <108>
3090 IF XX<0 THEN XX=(256*256)+XX              <059>
3100 RETURN                                    <110>
3103 :                                         <031>
3105 REM ************************              <191>
3110 REM *****  LADE SCREEN  *****             <122>
3115 REM ************************              <201>
3118 :                                         <046>
3120 OPEN 1,8,15                               <240>
3130 OPEN 2,8,2,"SCR"+STR$(SC)+",S,R"          <184>
3140 INPUT#1,A                                 <225>
3150 IF A<>0 THEN CLOSE 2:CLOSE 1:FOR I=0
     TO 24:SC$(I)="":NEXT I:RETURN             <215>
3160 FOR ZE=0 TO 24:B$=""                      <021>
3170 POKE 251,2:SYS 830                        <249>
3180 FOR I=512 TO 600:X=PEEK(I):IF X THEN
     B$=B$+CHR$(X):NEXT I                      <142>
3190 SC$(ZE)=B$                                <044>
3200 NEXT ZE                                   <200>
3210 CLOSE 2:CLOSE 1:RETURN                    <099>
3213 :                                         <141>
3215 REM *************************             <207>
3220 REM *****  SAVE  SCREEN  *****            <055>
3225 REM *************************             <217>
3228 :                                         <156>
3230 OPEN 1,8,2,"@:SCR"+STR$(SC)+",S,W"        <013>
3240 FOR ZE=0 TO 24                            <058>
3250 PRINT#1,SC$(ZE)                           <253>
3260 NEXT ZE                                   <004>
3270 CLOSE 1:ZE$="":PRINT"{CLR}";:RETURN       <068>
3273 :                                         <201>
3275 REM **********************                <112>
3280 REM **** EDIT A SCREEN ****               <079>
3285 REM **********************                <122>
3288 :                                         <216>
3290 GOSUB 3400                                <022>
3300 PRINT"{HOME}";:COMP=1                     <188>
3310 GOSUB 2750                                <242>
3315 IF LEFT$(ZE$,1)="N" THEN GOSUB 2630:G
     OSUB 2630:SC=VAL(BE$):GOSUB 3420:GOTO
     3300                                      <046>
3320 IF LEFT$(ZE$,1)="E" THEN ZE$="":COMP=
     0:GOTO 3220                               <152>
3321 IF LEFT$(ZE$,1)<>"I" THEN 3330            <187>
3322 GOSUB 2630:GOSUB 2630:Z=VAL(BE$):IF Z
     <0 OR Z>23 THEN GOSUB 3420:GOTO 3300      <242>
3323 GOSUB 2630:A=VAL(BE$):IF A<0 OR A>23
     THEN GOSUB 3420:GOTO 3300                 <033>
3324 FOR I=22-A TO Z STEP-1:SC$(I+A)=SC$(I
     ):SC$(I)="":NEXT                          <070>
3325 GOSUB 3420:GOTO 3300                      <213>
3330 IF LEFT$(ZE$,1)="S" THEN ZE$="":PRINT
     "{CLR}";:COMP=0:RETURN                    <147>
3331 IF LEFT$(ZE$,1)<>"D" THEN 3337            <246>
3332 GOSUB 2630:GOSUB 2630:Z=VAL(BE$):IF Z
     <0 OR Z>23 THEN GOSUB 3420:GOTO 3300      <254>
3333 GOSUB 2630:A=VAL(BE$):IF A<0 OR A>23
     THEN GOSUB 3420:GOTO 3300                 <045>
3334 FOR I=Z TO 22-A:SC$(I)=SC$(I+A):SC$(I
     +A)="":NEXT                               <131>
3335 GOSUB 3420:GOTO 3300                      <225>
3337 IF LEFT$(ZE$,1)="L" THEN GOSUB 3420:G
     OTO 3300                                  <128>
3340 ZE=VAL(ZE$)                               <004>
3350 ZE$=MID$(ZE$,3)                           <078>
3360 IF ZE>9 THEN ZE$=MID$(ZE$,2)              <032>
3370 SC$(ZE)=ZE$                               <141>
3380 GOSUB 2630:IF BE$="-->" THEN GOTO 322
     0                                         <037>
3390 GOTO 3310                                 <124>
3393 :                                         <067>
3395 REM ************************              <227>
3400 REM *****  LIST SCREEN  *****             <098>
3405 REM ************************              <237>
3408 :                                         <082>
3410 GOSUB 3110                                <128>
3420 PRINT"{CLR}";                             <056>
3430 FOR ZE=0 TO 23                            <248>
3440 PRINT RIGHT$(STR$(ZE),2);":";             <157>
3450 PRINT LEFT$(SC$(ZE),38)                   <018>
3460 NEXT ZE:RETURN                            <184>
3463 :                                         <137>
3465 REM **********************                <048>
3470 REM ** HOLE WERT VOM TOS **               <047>
3475 REM **********************                <058>
3480 AD=52992+PEEK(2)                          <065>
3490 X=PEEK(AD-1)+256*PEEK(AD-2)               <138>
3500 POKE 2,PEEK(2)-2:RETURN                   <243>
3503 :                                         <177>
3505 REM **********************                <088>
3510 REM ***    SAVE-SYSTEM    ***             <056>
3515 REM **********************                <098>
3518 :                                         <192>
3520 GOSUB 2630                                <118>
3530 OPEN 1,8,15,"S:"+BE$+".*":CLOSE 1         <206>
3540 OPEN 2,8,2,BE$+".VOC,P,W"                 <175>
3550 PRINT#2,AN:PRINT#2,VOC                    <063>
3560 FOR ZE=39 TO AN                           <075>
3570 PRINT#2,WO$(ZE)                           <202>
3580 PRINT#2,AD(ZE)                            <160>
3590 NEXT ZE                                   <082>
3600 CLOSE 2:BE$=BE$+".CODE"                   <016>
3610 POKE 187,FN L(720):POKE 188,FN H(720)     <013>
3620 FOR I=1 TO LEN(BE$)                       <095>
3630 POKE 719+I,ASC(MID$(BE$,I,1))             <115>
3640 NEXT I:POKE 183,LEN(BE$)                  <233>
3650 POKE 186,8:POKE 185,1                     <199>
3660 POKE 251,FN L(BE):POKE 252,FN H(BE)       <173>
3670 POKE 780,251                              <205>
3680 POKE 781,FN L(VOC)                        <225>
3690 POKE 782,FN H(VOC)                        <007>
3700 SYS 216+256*255                           <020>
3710 GOTO 700                                  <130>
3713 :                                         <133>
3715 REM **************************            <012>
3720 REM ****     LOAD SYSTEM     ****         <056>
3725 REM **************************            <022>
3728 :                                         <148>
3730 GOSUB 2630                                <074>
3740 OPEN 2,8,2,BE$+".VOC,P,R"                 <217>
3750 INPUT#2,AN,VOC                            <249>
3760 FOR ZE=39 TO AN                           <021>
3770 INPUT#2,WO$(ZE)                           <068>
3780 INPUT#2,AD(ZE)                            <026>
3790 NEXT ZE:CLOSE 2                           <162>
3800 SYS 50139,BE$+".CODE",8                   <110>
3810 GOTO 700                                  <230>
3813 :                                         <233>
3815 REM **************************            <112>
3820 REM ***         DATA          ***         <160>
3825 REM **************************            <122>
3828 :                                         <248>
3830 DATA 166,251, 32,198,255,160,  0, 32,
     207,255,201, 13,240,  7,153,  0           <161>
3840 DATA  2,200, 76, 69,  3,169,  0,153,
      0,  2, 76,204,255                        <248>
3850 FOR I= 830 TO 858:READ A:POKE I,A:Z=Z
     +A:NEXT I                                 <157>
3860 IF Z<>3379 THEN PRINT"{RVSON}FEHLER I
     N DATA{RVOFF}":END                        <107>
3870 RETURN                                    <118>
```

© 64'er

Listing 1. Tiny-Forth-Compiler (Schluß)

```
programm : vocabulary           c006 c3ea
------------------------------------------
c006 : a9 93 4c d2 ff 20 3f c0   66
c00e : 85 62 86 63 a2 90 a5 62   dc
c016 : 30 09 a9 20 20 d2 ff 38   42
c01e : 4c d4 bd a9 2d 20 d2 ff   98
c026 : 18 4c d4 bd 85 fe 86 fd   b7
c02e : a6 02 a5 fe 9d 00 cf e8   09
c036 : a5 fd 9d 00 cf e8 86 02   a4
c03e : 60 a6 02 ca bd 00 cf 85   f1
c046 : fd ca bd 00 cf 85 fe 86   4a
c04e : 02 a5 fe a6 fd 60 4c 3f   4a
c056 : c0 20 3f c0 20 2a c0 4c   fd
c05e : 2e c0 20 3f c0 8d 00 c0   d6
c066 : 8e 01 c0 20 3f c0 8d 02   dd
c06e : c0 8e 03 c0 ad 00 c0 ae   89
c076 : 01 c0 20 2a c0 ad 02 c0   28
c07e : ae 03 c0 4c 2a c0 c6 02   2f
c086 : c6 02 20 3f c0 e6 02 e6   56
c08e : 02 e6 02 e6 02 4c 2a c0   0d
c096 : a6 fc ca bd 00 ce 85 fd   ad
c09e : ca bd 00 ce 85 fe 86 fc   85
c0a6 : a5 fe a6 fd 60 85 fe 86   6f
c0ae : fd a6 fc a5 fe 9d 00 ce   6d
c0b6 : e8 a5 fd 9d 00 ce e8 86   cb
c0be : fc 60 38 ad 00 c0 ed 02   70
c0c6 : c0 8d 04 c0 ad 01 c0 ed   28
c0ce : 03 c0 8d 05 c0 60 18 ad   00
c0d6 : 00 c0 6d 02 c0 8d 04 c0   dc
c0de : ad 01 c0 6d 03 c0 8d 05   60
c0e6 : c0 60 a9 0d 4c d2 ff 20   7e
c0ee : 3f c0 8e 00 c0 8d 01 c0   2f
c0f6 : 20 3f c0 8e 02 c0 8d 03   1a
c0fe : c0 4c c0 c0 20 ed c0 ad   fc
c106 : 04 c0 0d 05 c0 f0 06 a9   4d
c10e : 00 aa 4c 2a c0 a9 00 a2   5a
c116 : 01 4c 2a c0 20 ed c0 30   b5
c11e : 06 a9 00 aa 4c 2a c0 a9   bb
c126 : 00 a2 01 4c 2a c0 20 60   2b
c12e : c0 4c 1a c1 20 3f c0 05   dc
c136 : fd f0 06 a9 00 aa 4c 2a   3d
c13e : c0 a9 00 a2 01 4c 2a c0   c4
c146 : 20 3f c0 8e 00 c0 8d 01   46
c14e : c0 20 3f c0 8e 02 c0 8d   1d
c156 : 03 c0 60 20 46 c1 ad 00   ff
c15e : c0 2d 02 c0 85 fd ad 01   4e
c166 : c0 2d 03 c0 85 fe 4c 2e   74
c16e : c0 20 46 c1 ad 00 c0 0d   00
c176 : 02 c0 85 fd ad 01 c0 0d   f9
c17e : 03 c0 85 fe 4c 2e c0 20   9c
c186 : 46 c1 ad 00 c0 4d 02 c0   18
c18e : 85 fd ad 01 c0 4d 03 c0   a1
c196 : 85 fe 4c 2e c0 20 46 c1   1d
c19e : 20 d4 c0 ae 04 c0 ad 05   35
c1a6 : c0 4c 2a c0 20 60 c0 20   77
c1ae : 46 c1 20 c0 c0 ae 04 c0   08
c1b6 : ad 05 c0 4c 2a c0 20 46   55
c1be : c1 a0 00 8c 04 c0 8c 05   e3
c1c6 : c0 a0 10 0e 04 c0 2e 05   a5
c1ce : c0 2e 02 c0 2e 03 c0 90   5d
c1d6 : 1d 18 ad 04 c0 6d 00 c0   e4
c1de : 8d 04 c0 ad 05 c0 6d 01   61
c1e6 : c0 8d 05 c0 90 08 ee 02   cf
c1ee : c0 d0 03 ee 03 c0 88 d0   af
c1f6 : d2 ae 04 c0 ad 05 c0 4c   d7
c1fe : 2a c0 20 46 c1 a0 10 a9   0e
c206 : 00 8d 04 c0 8d 05 c0 2e   46
c20e : 02 c0 2e 03 c0 2e 04 c0   6b
c216 : 2e 05 c0 38 ad 04 c0 ed   d8
c21e : 00 c0 aa ad 05 c0 ed 01   ef
c226 : c0 90 06 8d 05 c0 8e 04   fa
c22e : c0 88 d0 db 2e 02 c0 2e   34
c236 : 03 c0 60 20 00 c2 ae 02   8a
c23e : c0 ad 03 c0 4c 2a c0 20   07
c246 : 00 c2 ae 04 c0 ad 05 c0   e2
c24e : 4c 2a c0 20 96 c0 4c 2a   d8
c256 : c0 20 3f c0 4c ab c0 20   74
c25e : 60 c0 20 57 c2 4c 57 c2   83
c266 : 20 51 c2 e6 fc e6 fc 60   78
c26e : 20 51 c2 20 57 c0 20 51   8a
c276 : c2 20 60 c0 20 ed c0 30   4d
c27e : 08 20 54 c0 20 54 c0 38   dc
c286 : 60 e6 fc e6 fc 20 9b c1   38
c28e : 20 57 c2 18 60 a9 00 a2   a6
c296 : 01 20 2a c0 4c 6e c2 20   cd
c29e : 3f c0 e6 02 e6 02 a5 fe   4a
c2a6 : 10 c6 20 51 c2 20 57 c0   58
c2ae : 20 51 c2 4c 7a c2 20 3f   6e
c2b6 : c0 05 fd f0 02 38 60 18   2a
c2be : 60 20 3f c0 8a 4c d2 ff   6d
c2c6 : 20 87 ea e0 ff f0 04 e0   da
c2ce : 0d b0 02 a2 00 a9 00 4c   ee
c2d6 : 2a c0 20 c6 c2 20 57 c0   4d
c2de : 20 b4 c2 b0 06 20 54 c0   53
c2e6 : 4c d8 c2 60 86 fd 85 fe   c7
c2ee : a0 00 b1 fd f0 06 20 d2   20
c2f6 : ff c8 d0 f6 60 20 3f c0   f2
c2fe : 86 f7 20 3f c0 a0 00 b1   e4
c306 : fd 20 d2 ff c8 c4 f7 d0   fc
c30e : f6 60 20 3f c0 86 f9 20   8d
c316 : 3f c0 86 f7 85 f8 a2 00   00
c31e : a0 00 20 cf ff 91 f7 c9   c0
c326 : 0d f0 06 e8 c8 c4 f9 d0   86
c32e : f1 60 a5 02 4a aa a9 00   9a
c336 : 4c 2a c0 20 3f c0 86 f7   cf
c33e : 85 f8 20 3f c0 a0 01 91   68
c346 : f7 8a 88 91 f7 60 20 3f   58
c34e : c0 86 f7 85 f8 20 3f c0   0f
c356 : a0 00 8a 91 f7 60 20 3f   4d
c35e : c0 86 f7 85 f8 a0 [illegible] [illegible]   08
c366 : f7 aa c8 b1 f7 4c 2a c0   27
c36e : 20 3f c0 86 f7 85 f8 a0   ff
c376 : 00 b1 f7 aa a9 00 4c 2a   c2
c37e : c0 20 3f c0 ad 0c 03 ae   db
c386 : 0d 03 ac 0e 03 6c fd 00   8d
c38e : 20 3f c0 a4 02 8c 00 c0   19
c396 : 8a 0a 85 02 20 3f c0 ac   1f
c39e : 00 c0 84 02 4c 2a c0 20   b9
c3a6 : 3f c0 8e 00 c0 8d 01 c0   e7
c3ae : 20 3f c0 8e 02 c0 8d 03   d2
c3b6 : c0 20 3f c0 8e 04 c0 8d   95
c3be : 05 c0 ae 02 c0 ad 03 c0   16
c3c6 : 20 2a c0 ae 00 c0 ad 01   c0
c3ce : c0 20 2a c0 ae 04 c0 ad   aa
c3d6 : 05 c0 4c 2a c0 20 fd ae   f6
c3de : 20 d4 e1 a9 01 85 b9 a9   8c
c3e6 : 00 4c d5 ff 00 00 00 00   82
```

Listing 2. Maschinenspracheteil von Forth (Bitte beachten Sie die Eingabehinweise auf Seite 53)

```
 0: ( *** Zusatz-Befehle 0 *** )
 1:
 2:
 3: ( Processor-Register )
 4:
 5: 780 constant a-reg
 6: 781 constant x-reg
 7: 782 constant y-reg
 8: ( ------------------- )
 9:
10: ( Die folgenden Befehle )
11:
12: ( sollen zeigen,dass es )
13:
14: ( mit dem recht beschei- )
15:
16: ( denen Grundvokabular )
17:
18: ( moeglich ist,doch ein )
19:
20: ( sinnvolles,zweckmaess- )
21:
22: ( iges zu erstellen. )
23:-->
```

Listing 3. Zusatz-Befehle, Screen 0

```
 0:( *** zusatz-befehle 1 ***)
 1:
 2:
 3: (  2. Schleifenindex )
 4:
 5: : j
 6:     r> r> r>
 7:     dup 704 !
 8:     >r >r >r
 9:     704 @
10: ;
11:
12:
13:
14: (  Verlasse Schleife )
15:
16: : leave
17:     r> r>
18:     drop dup
19:     >r >r
20: ;
21:
22:
23: -->
```

Listing 4. Zusatz-Befehle, Screen 1

```
 0: ( *** Zusatz-Befehle 2 *** )
 1:
 2:
 3: ( open (addr l log nr sec --) )
 4:
 5: : open
 6:     185 c!   186 c! 184 c!
 7:     183 c!   187 !
 8:     65472 call
 9: ;
10:
11:
12:
13: ( close (log --) )
14:
15: : close
16:     cr 65484 call
17:     a-reg c!
18:     65475 call
19: ;
20:
21:
22:
23: -->
```

Listing 5. Zusatz-Befehle, Screen 2

```
 0: ( *** Zusatz-Befehele 3 *** )
 1:
 2:
 3: ( Ausgabe auf File (log --) )
 4:
 5: : out
 6:     x-reg c!
 7:     65481 call
 8: ;
 9:
10:
11:
12: ( Eingabe von File (log --) )
13:
14: : in
15:     x-reg c!
16:     65478 call
17: ;
18:
19:
20:
21:
22:
23: -->
```

Listing 6. Zusatz-Befehle, Screen 3

```
 0: ( *** Zusatz-Befehle 4 *** )
 1:
 2:
 3: ( Ausgabe auf Drucker )
 4:
 5: : printer
 6:     0 0 4 4 7 open
 7:     4 out
 8:     cr
 9: ;
10:
11:
12: ( Ausgabe auf Display )
13:
14: : display
15:     cr
16:     4 close
17: ;
18:
19:
20:
21: ( * ende * )
22:
23:
```

Listing 7. Zusatz-Befehle, Screen 4

Hi-Text

Hi-Text ist als Programmbaustein zur Ergänzung von Grafikerweiterungen gedacht. Mit ihm lassen sich die im Zeichensatzspeicher vorhandenen 512 Zeichen im Quasi-Direktmodus vor, während oder nach der Erstellung einer Hires-Grafik darstellen.

Das Programm entstand aus folgender Überlegung: Zum Beschriften von Grafikbildern wäre ein Auswählen der Zeichen, der Farbe, des Darstellungsmodus und natürlich auch der Position beim »Schreiben« sehr viel geeigneter als beim Erstellen des Programms. Allein schon wegen der dadurch möglichen sofortigen Kontrolle und eventuellen Korrektur.

Ich glaube, daß alle, die mit Hires-Grafik umgehen, von dem Einsatz des Programms profitieren können.

Kurzbeschreibung: Hi-Text ist ein Programmodul zur Darstellung der im Char-ROM des C 64 vorhandenen 512 Zeichen auf dem Hires-Bildschirm. Eine Besonderheit gebenüber den aus diversen Befehlserweiterungen bekannten »TEXT«-Befehlen ist die Tatsache, daß die Zeichen nach Aufruf des Programms über die Tastatur quasi im Direktmodus eingegeben werden, nachdem Farbe, Darstellungsmodus und Position direkt gewählt wurden. Das Programm hat eine Länge von 542 Byte. Da Hi-Text die Startadressen von Bit-Map und VideoRAM selbsttätig errechnet, kann es auch mit Grafikerweiterungen zusammenarbeiten, die mehrere Grafikseiten verwenden.

Einsatzmöglichkeiten

Das Programm ist bewußt als Programmodul konzipiert, das heißt, es soll zu allen Grafikerweiterungen passen. Dazu ist es notwendig, daß es ohne allzu großen Aufwand verschiebbar ist, denn nicht bei jeder Grafikerweiterung kann das Programm ab Adresse $C000 bleiben (zum Beispiel Turtle-Grafik im 64'er, Ausgabe 11/84). Assemblerprogrammierer können das Programm durch Ändern der Startadresse im Quellprogramm in einen ihnen passenden Bereich verschieben. Um jedoch auch allen anderen eine freie Verwendung zu ermöglichen, habe ich das Hilfsprogramm (Listing 3) geschrieben. Es liest das Programm von Diskette ein, legt es am oberen Ende des freien Basic-Speichers ab und schützt es vor dem Überschreiben durch Basic. Das Programm wurde mit verschiedenen Basic-Erweiterungen getestet. Es arbeitete stets einwandfrei.

Während der Entstehung des Programms wurde Hires 3.1 (aus der 64'er, Ausgaben 2 und 3/85) veröffentlicht. Bei dieser Grafikerweiterung gibt es einen ungenutzten Bereich ab Adresse $89B8, der jedesmal mitabgespeichert wird. Hier bietet sich ein Einbau von »Hi-Text« geradezu an. Der erste Teil (Berechnung der Basisadressen) kann natürlich entfallen; es genügt die High-Bytes der Startadressen von Bitmap ($A0) und VideoRAM ($8C) direkt zu übergeben. Der Einfachheit halber ist das Programm »Hi-Text 2.0« (Listing 1) schon auf diesen Bereich abgestimmt. Eingebaut wird das Programm wie folgt:

Hi-Text 2.0 mit dem MSE eingeben und abspeichern,
SMON Laden.
Hires 3.1 Laden.HI-TEXT 2.0 laden.
Mit SMON die erweiterte Version Hires3.2 speichern. Aufgerufen wird das neue Unterprogramm mit SYS 35256. Der Vollständigkeit halber sei hinzugefügt, daß diese Version von Hi-Text den Bereich von $89B8-$8BD5 belegt und die Adressen bis $8BDD als interne Speicher benutzt. Ein kleines Programm (Listing 2) demonstriert die Wirkung von Hi-Text in Verbindung mit Hires-3.

Umgang mit dem Programm

Aufruf durch SYS Adresse .»Schreiben« wie im Direktmodus. Zum Beenden »RETURN« drücken (wichtig vor Hardcopies, damit der Cursor verschwindet!)

Und nun viel Spaß beim Schreiben und Zeichnen in der hochauflösenden Grafik! (W. Gachot/gk)

Leserservice-Diskette

Auf der zu jeder 64'er-Ausgabe lieferbaren Diskette befinden sich auch die hier veröffentlichten Programme (Listing 1 bis 3) und Hires-3 mit integriertem Hi-Text. Laden Sie dann also nur Hires3.2 (mit ...,8,1) und anschließend DEMO (..,8). Mit RUN starten. Nachdem sich der Cursor meldet, können Sie über die Tastatur beliebigen Text eingeben.

```
10 REM *** DEMO ***                              <185>
20 REM HI-TEXT IST INTEGRIERT                    <135>
30 REM AUFRUF MIT SYS 35256                      <083>
40 POKE 56,128:POKE 52,128:SYS 37498             <094>
50 HFL,7,6                                       <231>
60 FOR I=20 TO 300 STEP 10                       <020>
70 LIN,I,0,160,199                               <056>
80 LIN,160,0,I,199                               <126>
90 NEXT                                          <100>
100 SYS 35256                                    <120>
110 PAU,5                                        <224>
120 HOF                                          <119>
```

© 64'er

Listing 2. In Verbindung mit Hires-3 läuft dieses Demo-Programm.
(Beachten Sie bitte die Eingabehinweise auf Seite 53.)

```
10 REM HILFSPROGRAMM ZUM VERSCHIEBEN VON H
   I-TEXT                                        <131>
20 REM ERST DAS GRAFIKPROGRAMM LADEN UND E
   VTL. VOR DEM UEBERSCHREIBEN                   <064>
30 REM DURCH BASIC SCHUETZEN!                    <101>
40 PRINT CHR$(147):PRINT                         <031>
50 PRINT"DISKETTE MIT 'HI-TEXT' EINLEGEN         <249>
60 PRINT                                         <162>
70 PRINT"DANN RETURN DRUECKEN!"                  <158>
80 GET T$:IF T$<>CHR$(13) THEN 80                <123>
90 SA=PEEK(55)+256*PEEK(56)-560                  <247>
100 HS=INT(SA/256):LS=SA-HS *256                 <108>
110 POKE 55,LS :POKE 56,HS                       <150>
120 POKE 51,LS :POKE 52,HS                       <064>
130 OPEN 1,8,2,"HI-TEXT"                         <194>
140 GET#1,A$:IF A$="" THEN A$=CHR$(0)            <201>
150 GET#1,B$:IF B$="" THEN B$=CHR$(0)            <024>
160 FOR I=0 TO 542                               <124>
170 GET#1,A$:IF A$="" THEN A$=CHR$(0)            <231>
180 A=ASC(A$)                                    <074>
190 POKE SA+I,A                                  <200>
200 NEXT                                         <210>
210 CLOSE 1                                      <221>
300 REM AENDERUNG DER SPRUNGADRESSEN             <181>
310 I =SA+515:IH=INT(I/256):IL=I-IH*256          <063>
320 E =SA+ 85:EH=INT(E/256):EL=E-EH*256          <011>
330 S =SA+ 76:SH=INT(S/256):SL=S-SH*256          <027>
340 R =SA+275:RH=INT(R/256):RL=R-RH*256          <145>
360 POKE SA+ 77,IL:POKE SA+ 78,IH                <160>
370 POKE SA+120,IL:POKE SA+121,IH                <172>
380 POKE SA+132,IL:POKE SA+133,IH                <076>
390 POKE SA+144,IL:POKE SA+145,IH                <237>
400 POKE SA+135,EL:POKE SA+136,EH                <234>
410 POKE SA+322,SL:POKE SA+323,SH                <211>
420 POKE SA+434,SL:POKE SA+435,SH                <182>
430 POKE SA+513,RL:POKE SA+514,RH                <204>
450 PRINT CHR$(147)                              <225>
460 PRINT"HI-TEXT IST JETZT IM SPEICHER!"        <108>
470 PRINT                                        <062>
480 PRINT"AUFRUF MIT SYS";SA                     <206>
490 NEW                                          <118>
```

© 64'er

Listing 3. Hi-Text kann mit diesem Programm verschoben werden, aber auch natürlich mit dem SMON

```
programm : hi-text 2.0  89b8 8bd6
---------------------------------
89b8 : a9 a0 85 a8 a9 8c 85 a6  8a
89c0 : a9 00 85 cc 85 d3 85 d6  1f
89c8 : 8d d6 8b 85 a5 85 a7 a8  ca
89d0 : a9 ff 8d d8 8b a9 d0 8d  5c
89d8 : db 8b b1 a5 4a 4a 4a 4a  4f
89e0 : 8d 86 02 20 b8 8b ad 86  e1
89e8 : 02 8d 20 d0 ad 18 d0 29  04
89f0 : 02 f0 0a ad db 8b 09 08  f1
89f8 : 8d db 8b d0 08 ad db 8b  e4
8a00 : 29 f7 8d db 8b 20 e4 ff  51
8a08 : d0 1e ad d6 8b f0 09 ad  ed
8a10 : d8 8b d0 03 20 b8 8b 60  f9
8a18 : a5 cf cd d7 8b f0 cd 8d  a6
8a20 : d7 8b 20 b8 8b 4c ec 89  be
8a28 : 8d d9 8b ad d8 8b d0 03  6d
8a30 : 20 b8 8b ad d9 8b 20 cb  57
8a38 : e8 c9 91 d0 20 a5 d6 f0  f0
8a40 : 1c c6 d6 38 a5 a7 e9 40  3c
8a48 : 85 a7 a5 a8 e9 01 85 a8  2d
8a50 : 38 a5 a5 e9 28 85 a5 a5  92
8a58 : a6 e9 00 85 a6 c9 11 d0  42
8a60 : 22 a5 d6 c9 18 f0 1c e6  8b
8a68 : d6 18 a5 a7 69 40 85 a7  a7
8a70 : a5 a8 69 01 85 a8 18 a5  2d
8a78 : a5 69 28 85 a5 a5 a6 69  82
8a80 : 00 85 a6 c9 9d d0 2a a5  7a
8a88 : d3 d0 17 a5 d6 f0 22 a9  [illegible]
8a90 : 28 85 d3 c6 d6 38 a5 a5  59
8a98 : e9 28 85 a5 a5 a6 e9 00  e3
8aa0 : 85 a6 38 a5 a7 e9 08 85  30
8aa8 : a7 a5 a8 e9 00 85 a8 c6  e5
8ab0 : d3 c9 1d d0 31 a5 d3 c9  ec
8ab8 : 27 90 19 a5 d6 c9 18 f0  20
8ac0 : 22 a9 ff 85 d3 e6 d6 18  67
8ac8 : a5 a5 69 28 85 a5 a5 a6  09
8ad0 : 69 00 85 a6 18 a5 a7 69  90
8ad8 : 08 85 a7 a5 a8 69 00 85  22
8ae0 : a8 e6 d3 4c e3 89 c9 12  50
8ae8 : d0 08 ad db 8b 09 04 8d  cf
8af0 : db 8b c9 92 d0 08 ad db  11
8af8 : 8b 29 fb 8d db 8b c9 13  30
8b00 : d0 12 a9 00 85 d3 85 d6  fe
8b08 : 85 a5 85 a7 a9 a0 85 a8  bd
8b10 : a9 8c 85 a6 c9 0d d0 05  88
8b18 : ee d6 8b e6 cc ad d9 8b  ea
8b20 : 18 c9 ff 90 04 a9 5e b0  97
8b28 : 33 c9 c0 90 04 29 7f b0  6b
8b30 : 2b c9 a0 90 06 29 7f 09  34
8b38 : 40 b0 21 c9 80 90 02 b0  48
8b40 : 18 c9 60 90 04 29 5f b0  cf
8b48 : 13 c9 40 90 04 29 1f b0  c9
8b50 : 0b c9 20 90 04 29 3f b0  42
8b58 : 03 4c e3 89 a2 00 8e dd  cb
8b60 : 8b 8d dc 8b 18 0e dc 8b  d7
8b68 : 2e dd 8b 0e dc 8b 2e dd  c8
8b70 : 8b 0e dc 8b 2e dd 8b 18  db
8b78 : a9 00 6d dc 8b 85 a9 ad  ff
8b80 : db 8b 6d dd 8b 85 aa 78  b8
8b88 : a5 01 29 fb 85 01 a0 07  69
8b90 : b1 a9 91 a7 88 10 f9 a5  ab
8b98 : 01 09 04 85 01 58 ad 86  66
8ba0 : 02 0a 0a 0a 0a 8d da 8b  fb
8ba8 : a4 d3 b1 a5 29 0f 0d da  4c
8bb0 : 8b 91 a5 a9 1d 4c b5 8a  c2
8bb8 : a0 07 78 a9 35 85 01 b1  16
8bc0 : a7 49 ff 91 a7 88 10 f7  2d
8bc8 : a9 37 85 01 58 ad d8 8b  fc
8bd0 : 49 ff 8d d8 8b 60 ff 00  53
```

Listing 1. Hi-Text 2.0 kann in Hires-3 eingebaut werden, aber auch in andere Basic-Erweiterungen. (Beachten Sie bitte Eingabehinweise auf Seite 54.)

Hier gibt's Clubs

Computer-Club Eifel G. o. E.,
c/o Walter Harth,
An den Hofwiesen 3,
B-4750 Bütgenbach/Belgien,
Club-Treffen alle 14 Tage, Programme entwerfen, Veranstaltungen, Ausstellungen, Feriencamps;

@@-Computer-Club,
z. Hd. Wolfgang Petzold,
Westendstr. 19, 6053 Obertshausen 1, Tel. 06104/44181 (werktags: 14.45 — 16.15 Uhr, Sonn- und Feiertags: 14.00 — 18.00 Uhr), monatliche Clubzeitschrift, Tips- und Erfahrungsaustausch, Hardware- und Basteleientips, Mailbox in Vorbereitung;

F-V-C Hünenburg,
Postfach 140334, Kennw. »INFO«,
4800 Bielefeld 14,
eigener Clubraum mit Bar-Atmosphäre, jeden Mittwoch ab 19.00 Uhr, Clubeigene Hardware (VC 20, C 64, 1530, 1541), Informationsaustausch, Basic, Hardware-Veränderungen und -Erweiterungen, Informationsbeschaffung bei Neuheiten, Geselligkeiten;

C=64/Commodore Interessengemeinschaft Chiemgau/Inn, e. V.,
Postfach 1207,
D 8090 Wasserburg,
Wissen vermitteln, Informationsaustausch, Hilfe bei Problemen, Tips zum Kauf von Geräten, Kurse für Basic, Assembler und andere Programmiersprachen, Treffen 14-tägig, Samstags;

Computer-Club Husum,
Eckhard Schiffler, Pellwormer Straße 6, 2250 Husum,
alle Homecomputer, kostenlose Basic-Kurse und Maschinensprache, Clubbeitrag zwei bis drei Mark, Clubmailbox (04841-1881), Zeitschrift geplant, Programmbibliothek (keine Raubkopien erwünscht);

Commodores Nordheide,
Heuweg 10, 2110 Buchholz,
Clubzeitschrift, Softwarebibliothek, Problemlösungen geplant, Clubtreffen, u.a.;

IC Computerclub,
Beethovenstr. 66, 4815 Schloß Holte-Stuk., Tel. 05207/87700,
wir suchen alle Computerfreaks, die bereit sind, folgende Dienste kostenlos in Anspruch zu nehmen beziehungsweise zu teilen: Softwaretausch, Entwicklung von Hard- u. Software, Beratung bei Problemen, Reparaturen, Programmierkurse, Erstellung einer Clubzeitung, alle Systeme/Computertypen, Anfragen, wenn möglich, bitte mit Rückporto;

Arnbacher Softy-Club,
Schwarzwaldstraße 11,
7540 Neuenbürg-Arnbach,
Clubzeitung, Treffs, Erfahrungsaustausch, Listings abtippen, Softwareaustausch und Produktion, Simons Basic, etc.;

Arbeitsgemeinschaft der C 64-User,
Robert Klima, Birkenweg 7,
8901 Emersacker,
überregionale Arbeitsgemeinschaft, Informationsaustausch ausschließlich in Clubzeitschrift »Matrix 64«, Beiträge werden honoriert;

Der CCD (Computer Club Dietzenbach),
Frank Siering, Rosenweg 23,
6057 Dietzenbach
Austausch von Soft-, Hardware und Erfahrungen, Clubzeitschrift ist geplant, möchte auch Kontakte zu anderen Clubs, Informationen gegen Rückporto;

1. Computer-Club Pforzheim/Enzkreis e. V.,
Rüdiger Goetsch, Schulerstr. 2,
7530 Pforzheim-Büchenbronn,
Tel. 07231/71903,
vom programmierbaren Taschenrechner bis zur Groß-EDV-Anlage ist alles vertreten, Einsteiger, Programmersteller und betriebliche Anwender treffen sich an Clubabenden;

Schach dem C 64!

Eines der interessantesten Gebiete der Computertechnik ist sicherlich die Beschäftigung mit »künstlicher Intelligenz«. Um erste Erfahrungen auf diesem Gebiet zu sammeln, bietet sich ein Schachprogramm gut an. Hier ist eines zum Abtippen!

Man erkennt auch sofort, daß es an diesem Programm keine »künstliche Intelligenz« im eigentlichen Sinne gibt. Die »Intelligenz« beruht nur auf festen, mathematischen Formeln. Trotzdem ist es interessant, auf diesem Gebiet zu experimentieren. Das vorliegende Programm ist ein voll funktionsfähiges Schachprogramm, das über eine grafische Spielfeldanzeige verfügt und eine sehr kurze Rechenzeit von unter 10 Sekunden hat. Natürlich ist das Programm sehr spielstark. Der Sinn dieses Programms ist denn auch ein anderer. Dieses Programm soll dem Benutzer ein Grundstein für selbstgeschriebene Erweiterungen sein. Der interessierte Anwender ist aufgefordert, kreativ zu werden und eigene Ergänzungen zu installieren. So bietet sich zum Beispiel das Stellungsbewertungsprogramm besonders dazu an, erweitert und verfeinert zu werden. Auch die Grafik kann sicher verbessert werden.

So arbeitet das Programm

Der Programmablauf sieht wie folgt aus: Nachdem das Spielfeld initialisiert und ausgegeben ist, wird eine Eingabe eines weißen Zugs erwartet, zum Beispiel E2E4. Der Zug wird auf seine Richtigkeit überprüft und, falls er korrekt ist, ausgeführt. Dann wird eine Schleife gestartet, die sämtliche Spielfelder auf schwarze Figuren hin untersucht. Wird eine schwarze Figur gefunden, so wird der Zuggenerator angesprungen. Hier werden die der Figur möglichen Züge ausgeführt und mittels der Stellungsbewertung beurteilt. Nachdem alle Figuren überprüft wurden, wird der schwarze Zug mit seiner Wertigkeit ausgegeben. Nach einer kurzen Warteschleife wird die aktuelle Stellung ausgegeben, und es kann der nächste weiße Zug ausgeführt werden.

Die verwendete Maschinenunterroutine liegt im Speicherbereich von 49408 ($C100) bis 49864 ($C2D0). Das Unterprogramm überprüft, ob der schwarze König von irgendeiner weißen Figur angegriffen ist. Falls dies der Fall ist, so wird der Wert 128 an Basic übergeben, andernfalls der Wert 0. Es ist nun möglich, mit dem MC-Programm zu überprüfen, ob eine beliebige schwarze Figur angegriffen ist. Dazu wird in Speicherstelle 49409 die Position der aktuellen Figur (im Bewertungsprogramm ZF) geschrieben und in die Speicherstelle 49414 der Wert der Figur (zum Beispiel 130 = Springer). Nach Aufruf des Unterprogramms müssen unbedingt (!!) wieder die Standardwerte gePOKEt werden (49409:100 / 49414:134). Wird dies nicht beachtet, so kann der Computer keine regelwidrigen Stellungen mehr erkennen und antwortet mit unlogischen Zügen. Das Maschinenprogramm ist somit ideal dazu geeignet, um in einem selbst ergänzten Stellungsbewertungs-Programm eingesetzt zu werden. Man sollte jedoch beachten, daß ein umfangreiches Bewertungsprogramm die Rechenzeit pro Zug nicht unerheblich steigert.

Um dem Computer das Schachspiel beizubringen, sind im wesentlichen drei Probleme zu lösen:

1. Die interne Darstellung von Spielfeld und Figuren.
2. Der Zuggenerator: Er muß die Figuren den Regeln entsprechend bewegen und eine Liste möglicher Züge generieren.
3. Die Stellungsbewertung. Von ihrer Genauigkeit hängt die Spielstärke des Programms ab.

Nachfolgend nun eine Anleitung zur Lösung der drei Teilprobleme.

Da der Computer nicht auf das Schachbrett sehen kann, muß man es intern in einer Computerform darstellen, mit der der Computer arbeiten kann. Es bietet sich an, jedem Feld des Schachbretts eine Speicherstelle im Computer zuzuordnen.

100 bis 520:	Dieses Programm dient zur Stellungsbewertung. Die neue Position wird aufgestellt (100) und die Position auf Rechtmäßigkeit geprüft (110; Überprüfung, ob König im Schach). Falls die Position nicht legal ist, wird die weitere Bewertung übergangen.
210 bis 220:	Hier wird überprüft, ob eine Figur auf ihrem Ausgangsfeld beziehungsweise ihrem vorgesehenen Zielfeld von einem Bauern angegriffen ist.
500 bis 520:	Falls der Wert der Stellung besser ist wie der BESTMOVE bisher, so wird der überprüfte Zug als bester Zug eingetragen. Bei gleicher Wertigkeit entscheidet der Zufallsgenerator (510).
1000 bis 1090:	Hier wird das Spielfeld aufgebaut. Näheres siehe Zeichnung des Spielfeldes. Außerdem wird der USER-Vektor für das MC-Unterprogramm, das die Stellungen auf Schachgebote testet, gesetzt.
2000 bis 2100:	Das Schachbrett mit Randbezeichnungen und der aktuellen Figurenstellung wird ausgegeben. Außerdem wird der Positionswert (BW) auf ein definiertes Minimum gesetzt.
3000 bis 3130:	Hier wird ein weißer Zug eingelesen und, wenn er auf seine Richtigkeit geprüft wurde, ausgeführt. Die Legalitätsprüfung kann noch erweitert werden.
4000 bis 4040:	Diese Schleife sucht die Positionen der schwarzen Figuren (Wert größer 128) und verzweigt bei gefundener Figur in den Zuggenerator.
5000 bis 5090:	Es wird der schwarze Zug und seine Wertigkeit ausgegeben. Besonders bei Erweiterungen der Stellungsbewertung ist diese Kontrolle wichtig, um zu sehen, ob der Computer den Wert des Zugs richtig einschätzt. Nach zirka 2 Sekunden wird wieder zur Spielfeldanzeige gesprungen.
6000 bis 6800:	Dies ist das Kernstück des Programms: der Zuggenerator. Hier wird geprüft, welche Figur gefunden wurde, und die möglichen Züge werden ausgeführt. Dabei wird bei jeder Stellung das Bewertungs-Unterprogramm aufgerufen. Anschließend wird wieder zur Suchschleife (4000-) verzweigt.

Tabelle 1. Die wichtigsten Routinen

AA	: Wert der aktuellen Stellung (128 = ungültig)
AF	: Ausgangsfeld, das vom Zuggenerator übergeben wird
BA	: Ausgangsfeld des bisher besten gefundenen Zuges
BZ	: Zielfeld des bisher besten gefundenen Zuges
W	: Figurenwert für die Spielfeldausgabe: 1 = Bauer, 2 = Springer, 3 = Läufer, 4 = Turm, 5 = Dame, 6 = König. Schwarze Figuren sind im Wert 128 höher.
WZ	: Wert des Zielfeldes. Wird gebraucht, da das Bewertungsprogramm den Inhalt des Zielfeldes überschreibt.
X	: Aktuelles Figurenfeld mit schwarzer Figur
ZF	: Zielfeld; wird vom Zuggenerator übergeben.

Tabelle 2. Variablenübersicht

Den Inhalt der entsprechenden Speicherstellen stellen die Figuren dar, die sich auf dem Schachbrett befinden. Nun taucht aber ein zweites Problem auf: Wie kann der Computer auf einfache Weise überprüfen, wann sich eine Figur am Spielfeldrand befindet und nicht weiterziehen kann. Hier bietet es sich an, daß man um das eigentliche Spielfeld noch einen Rand installiert, der einen speziellen Zahlenwert enthält (Bild 1). In diesem Fall signalisiert der Zahlenwert 128 dem Rechner, daß dieses Feld nicht zum eigentlichen Spielfeld gehört und er es somit mit seinen Figuren nicht betreten darf. Damit hätten wir das Schachbrett in einer computergerechten Form intern dargestellt. Als letztes Problem müssen noch die verschiedenen Spielfiguren auf dem internen Brett codiert werden. Dazu ordnet man jeder Figur eine Zahl zu. Die weißen Figuren bekommen die Werte 1 bis 6 und die schwarzen Figuren die Werte 1 bis 6 plus 128. Somit ist eine einfache Unterscheidung zwischen weißen und schwarzen Figuren möglich: Weiße Figuren haben einen Wert kleiner 128 und schwarze Figuren einen Wert größer 128. Leere Felder werden mit Null initialisiert. Sehen wir uns jetzt im Listing an, wie die Probleme innerhalb des Programms gelöst werden (Zeilen 1030 bis 1080). Das Spielfeld liegt im Rechner ab Speicherstelle 49152. In den Zeilen 1030 bis 1050 werden zunächst der Spielfeldrand (128) und die leeren Felder (0) initialisiert (vergleiche auch Bild 1). In Zeile 1060 werden dann die weißen (1) und schwarzen Bauern (1 + 128 = 129) auf die zweite beziehungsweise siebte Reihe des Spielfeldes positioniert. Die beiden Zeilen 1070 und 1080 dienen dazu, die restlichen Figuren auf den Grundreihen zu positionieren.

Der Zuggenerator

Doch mit dem Spielbeginn taucht ein neues Problem auf: Wie bringe ich den Computer dazu, daß er die Figuren entsprechend den Regeln bewegt? Als Beispiel wollen wir dies mit den Turmzügen durchsprechen (Zeilen 6440 bis 6560). Als Grundlage dient hier das Spielfeld in Bild 1. Der Turm darf den Regeln nach nur senkrecht und waagerecht ziehen, und zwar so lange, bis er auf den Spielfeldrand oder eine andere Figur trifft. Das Ziehen des Turmes bedeutet somit auf dem Computerspielfeld, daß zum Ausgangsfeld des Turmes die Vielfachen von 1 oder 10 addiert beziehungsweise substrahiert werden. Dieses Verfahren wird so lange wiederholt, bis der Turm auf ein Hindernis (Spielfeldrand oder andere Figur) trifft. Diese Zuggenerierung wird vom Programm wie folgt realisiert: Als Beispiel betrachten wir die Zuggenerierung senkrecht nach »oben«, also die Addition der Vielfachen von 10 zum Ausgangsfeld. Die anderen Zugrichtungen ergeben sich analog zu der betrachteten. In Zeile 6440 wird zunächst ein Flag auf Null gesetzt (FF=0). Dieses Flag wird immer dann auf 1 gesetzt, wenn der Turm bei seinem Zug auf ein Hindernis trifft. Als nächstes wird eine FOR-NEXT-Schleife gestartet, die es ermöglicht, alle Felder in »+10-Richtung« zu überprüfen. Die Variable X symbolisiert dabei das momentane Standfeld des Turmes, und Z durchläuft alle möglichen Zielfelder. Zum Schluß wird in Zeile 6440 geprüft, ob auf dem vorgesehenen Zielfeld bereits eine eigene Figur steht oder ob das Feld nicht mehr zum eigentlichen Spielfeld gehört. In beiden Fällen wird das Flag FF auf 1 gesetzt. In Zeile 6450 wird zunächst geprüft, ob das Flag FF gesetzt ist. Ist dies der Fall, so ist das Zielfeld Z regelwidrig, und das folgende wird übersprungen. Da FF jedoch in der gesamten Schleife nicht zurückgesetzt wird, wird sie bis zu ihrem Ende durchlaufen, ohne daß weitere Stellungen ausgewertet werden. Nehmen wir jetzt aber an, daß FF noch Null ist und die Stellung somit legal. Dann wird in Zeile 6450 das erwogene Zielfeld ZF gleich Z gesetzt und die Stellungsbewertung ab Zeile 100 aufgerufen. Anschließend wird noch geprüft, ob auf Feld Z eine weiße Figur steht. Diese Figur darf zwar geschlagen werden, verhindert aber den weiteren Vormarsch des Turmes in diese Richtung. Somit wird das Flag FF in diesem Falle auf 1 gesetzt. Ich nehme an, daß das Prinzip der Zuggenerierung jetzt deutlich geworden ist. Es läßt sich relativ einfach auf die anderen Figuren übertragen, wenn man ihre speziellen Gangarten berücksichtigt.

Die Stellungsbewertung

Somit bleibt als letztes Problem nur noch die Bewertung der entstehenden Stellungen (Zeilen 100 bis 520). Wozu überhaupt eine Stellungsbewertung? Jeder, der schon einmal Schach gespielt hat, weiß, daß es nicht genügt, nur die Regeln zu beherrschen. Man muß zwischen guten und schlechten Zügen unterscheiden können, wenn man das Spiel gewinnen will. Diese Aufgabe hat die Stellungsbewertung.

Versuchen wir, uns die Funktionsweise mit Hilfe des Listings zu verdeutlichen. Zeile 100 führt zunächst einmal den erwogenen Zug auf dem Spielfeld aus. Dazu muß der Inhalt des Zielfeldes in der Variablen WZ gesichert werden, da er sonst verloren gehen würde. Anschließend wird der Zug ausgeführt. Zeile 110 gehört noch nicht zum eigentlichen Stellungsbewertungs-Programm. Hier wird überprüft, ob in der entstandenen Stellung der schwarze König im Schach steht und die Stellung somit illegal ist. Dies wird dadurch angezeigt, daß das Maschinensprache-Programm der Variablen AA den Wert 128 übergibt. Ist dies der Fall, so wird die Stellungsbewertung übersprungen und nach Zeile 520 verzweigt, wo der ausgeführte Zug zurückgenommen und in den Zuggenerator gesprungen wird. In Zeile 200 wird der Wert der Stellung (AA) zunächst um den ursprünglichen Wert des Zielfeldes erhöht. Hat auf dem Zielfeld eine weiße Figur gestanden, so ist diese geschlagen, und der Zug ist somit höherwertig, als wenn

Das Spielfeld:

	110	111	112	113	114	115	116	117	118	119
	128	128	128	128	128	128	128	128	128	128
	100	101	102	103	104	105	106	107	108	109
	128	128	128	128	128	128	128	128	128	128
8	90	91	92	93	94	95	96	97	98	99
	128	132	130	131	133	134	131	130	132	128
7	80	81	82	83	84	85	86	87	88	89
	128	129	129	129	129	129	129	129	129	128
6	70	71	72	73	74	75	76	77	78	79
	128	0	0	0	0	0	0	0	0	128
5	60	61	62	63	64	65	66	67	68	69
	128	0	0	0	0	0	0	0	0	128
4	50	51	52	53	54	55	56	57	58	59
	128	0	0	0	0	0	0	0	0	128
3	40	41	42	43	44	45	46	47	48	49
	128	0	0	0	0	0	0	0	0	128
2	30	31	32	33	34	35	36	37	38	39
	128	1	1	1	1	1	1	1	1	128
1	20	21	22	23	24	25	26	27	28	29
	128	4	2	3	5	6	3	2	4	128
	10	11	12	13	14	15	16	17	18	19
	128	128	128	128	128	128	128	128	128	128
	0	1	2	3	4	5	6	7	8	9
	128	128	128	128	128	128	128	128	128	128
		A	B	C	D	E	F	G	H	

Bild 1. Die interne Spielfeld-darstellung

schwarzer Bauer: -10 beziehungsweise -9 oder -11 beim Schlagen
schwarzer Springer:
-21, -19, -12, -8, 8, 12, 19, 21
schwarzer Turm:
nx-10, nx10, nx-1, nx1
schwarzer Läufer:
nx-11, nx11, nx-9, nx9
schwarze Dame:
kombiniert Läufer- und Turmzüge
schwarzer König:
-11, -10, -9, -1, 1, 9, 10, 11

Bild 2. Die verschiedenen Zugmöglichkeiten

Schwarz auf ein leeres Feld (WZ=0) gezogen hätte. In Zeile 210 wird geprüft, ob die Figur auf ihrem Ausgangsfeld angegriffen war. Dies ist der Fall, wenn der Wert der Felder X-9 oder X-11 gleich 1 ist, was bedeutet, daß die Figur auf ihrem Ausgangsfeld bedroht ist. Ein Wegziehen ist somit zu empfehlen, und der Wert AA wird um den Figurenwert erhöht. Das gleiche wird für das Zielfeld überprüft. Ist die Figur dort angegriffen, so wird ihr Wert von AA abgezogen, um zu zeigen, daß das Zielfeld ungünstig gewählt ist. Falls der berechnete Zug eine höhere Wertigkeit hat als der bisher beste Zug (BW), wird der soeben berechnete Zug als bester Zug eingeordnet, und die Ausgangs- und Zielkoordinaten (BA und BZ) werden ebenso wie die bisher beste Wertigkeit (BW) aktualisiert. Danach wird aus Zeile 500 nach 520 gesprungen, wo das Spielfeld zurückgesetzt und das Bewertungsprogramm verlassen wird. Falls der aktuelle Zug exakt die gleiche Wertigkeit hat wie der bisher beste Zug, so entscheidet der Zufallsgenerator darüber, ob der neue Zug als bester eingelistet werden soll. Diese Einrichtung sorgt dafür, daß der Computer in einer bestimmten Stellung nicht immer die gleichen Züge macht, sondern ein Spiel variiert. Das Stellungsbewertungs-Programm ist für eigene erste Experimente besonders geeignet. Man kann zum Beispiel noch eigene Kriterien zur Stellungsbewertung einführen. Jedoch sollte man bedenken, daß das Bewertungsprogramm besonders zeitkritisch ist, da es pro Zugberechnung etwa 30- bis 50mal durchlaufen wird. Jede weitere Zeile macht sich also in der Rechenzeit des Computers bemerkbar.

(Bernd Bettermann/ev)

Literaturhinweis: Computerschach von Ludek und Vas I. Kühnmund, Heyne-Buch Nr. 4704, Wilhelm Heyne Verlag, München, Taschenbuch, Preis: zirka 5,80 Mark.
Dieses Buch bietet eine ausführliche und leicht verständliche Einführung in das Thema Computerschach. Außerdem beinhaltet es einige interessante Partien zwischen Mensch und Computer.

```
1 REM ****** SCHACHLADER C-64 ******       <055>
2 REM **     BY BERND BETTERMANN    **     <181>
3 REM **       AMELITHER STR.21     **     <060>
4 REM **       3417 BODENFELDE      **     <134>
5 REM ******************************       <055>
10 :                                       <242>
20 FOR I=49408 TO 49864:READ Q:POKE I,Q:NE
   XT                                      <132>
30 :                                       <006>
40 PRINT CHR$(147):PRINT:PRINT"LOAD";CHR$(
   34);"SCHACH";CHR$(34);",8"              <088>
50 PRINT:PRINT:PRINT:PRINT:PRINT"RUN";CHR$
   (19);                                   <060>
60 POKE 198,2:POKE 631,13:POKE 632,13:END  <218>
100 :                                      <076>
101 DATA 162,100,189,0,192,201,134,240,4,2
    02,208,246,96,134,20,169               <014>
102 DATA 192,133,21,160,0,138,56,233,9,133
    ,20,177,20,201,1,208                   <251>
103 DATA 1,96,138,56,233,11,133,20,177,20,
    201,1,208,1,96,138                     <162>
104 DATA 56,233,21,133,20,177,20,201,2,208
    ,1,96,138,56,233,19                    <040>
105 DATA 133,20,177,20,201,2,208,1,96,138,
    56,233,12,133,20,177                   <096>
106 DATA 20,201,2,208,1,96,138,56,233,8,13
    3,20,177,20,201,2                      <204>
107 DATA 208,1,96,138,24,105,8,133,20,177,
    20,201,2,208,1,96                      <129>
108 DATA 138,24,105,12,133,20,177,20,201,2
    ,208,1,96,138,24,105                   <019>
109 DATA 19,133,20,177,20,201,2,208,1,96,1
    38,24,105,21,133,20                    <174>
110 DATA 177,20,201,2,208,1,96,138,56,233,
    11,133,20,177,20,201                   <245>
111 DATA 6,208,1,96,138,56,233,10,133,20,1
    77,20,201,6,208,1                      <248>
112 DATA 96,138,56,233,9,133,20,177,20,201
    ,6,208,1,96,138,56                     <166>
113 DATA 233,1,133,20,177,20,201,6,208,1,9
    6,138,24,105,1,133                     <112>
114 DATA 20,177,20,201,6,208,1,96,138,24,1
    05,9,133,20,177,20                     <067>
115 DATA 201,6,208,1,96,138,24,105,10,133,
    20,177,20,201,6,208                    <054>
116 DATA 1,96,138,24,105,11,133,20,177,20,
    201,6,208,1,96,134                     <067>
117 DATA 2,134,20,165,20,24,105,10,133,20,
    177,20,201,4,208,1                     <148>
118 DATA 96,201,5,240,251,201,0,240,234,13
    4,20,165,20,56,233,10                  <040>
119 DATA 133,20,177,20,201,4,208,1,96,201,
    5,240,251,201,0,240                    <223>
120 DATA 234,134,20,134,20,165,20,24,105,1
    ,133,20,177,20,201,4                   <107>
121 DATA 208,1,96,201,5,240,251,201,0,240,
    234,134,20,165,20,56                   <035>
122 DATA 233,1,133,20,177,20,201,4,208,1,9
    6,201,5,240,251,201                    <000>
123 DATA 0,240,234,134,20,165,20,24,105,11
    ,133,20,177,20,201,3                   <066>
124 DATA 208,1,96,201,5,240,251,201,0,240,
    234,134,20,165,20,24                   <100>
125 DATA 105,9,133,20,177,20,201,3,208,1,9
    6,201,5,240,251,201                    <007>
126 DATA 0,240,234,134,20,165,20,56,233,9,
    133,20,177,20,201,3                    <146>
127 DATA 208,1,96,201,5,240,251,201,0,240,
    234,134,20,165,20,56                   <041>
128 DATA 233,11,133,20,177,20,201,3,208,1,
    96,201,5,240,251,201                   <224>
129 DATA 0,240,234,160,0,32,162,179,96     <051>

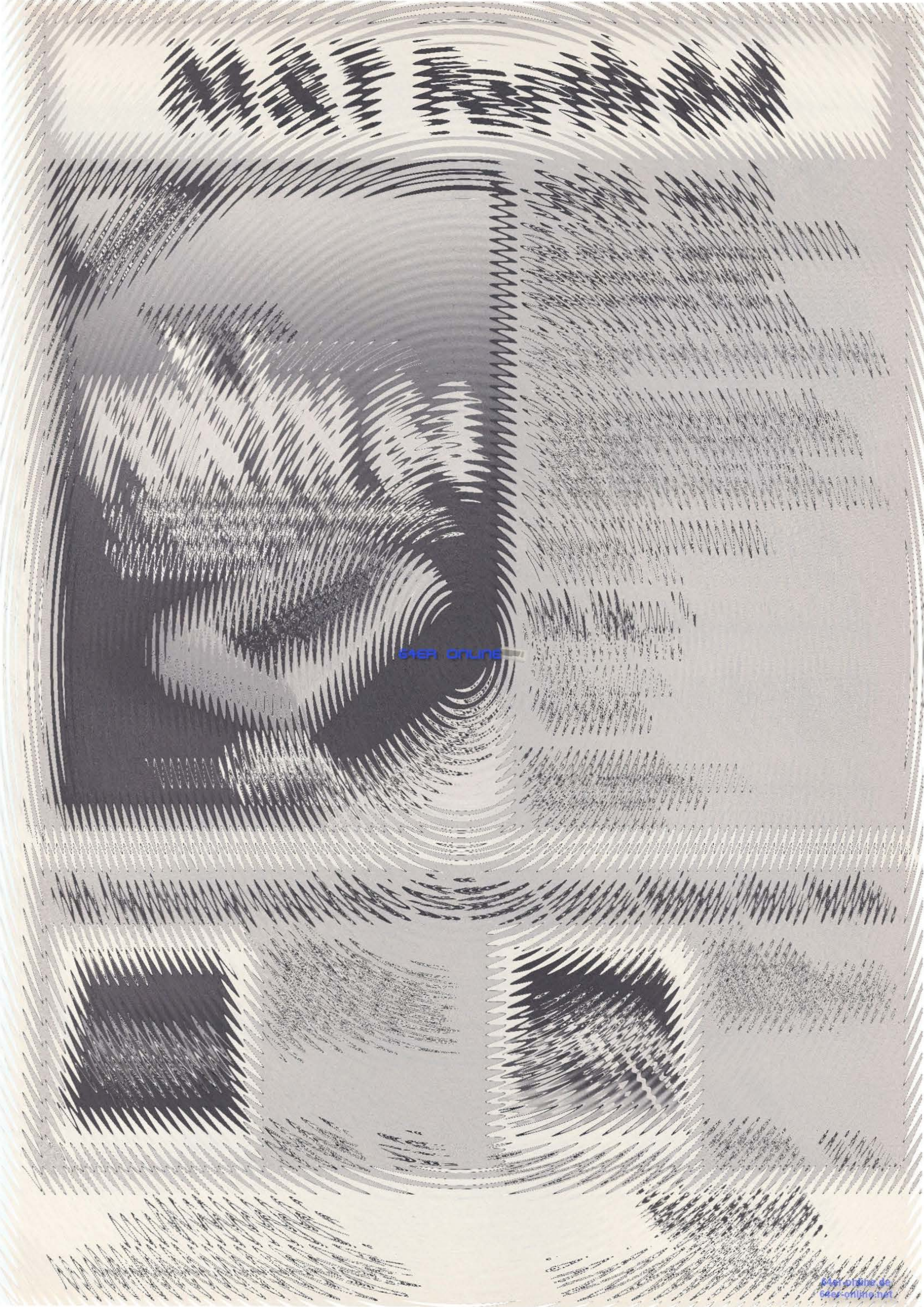
© 64'er
```

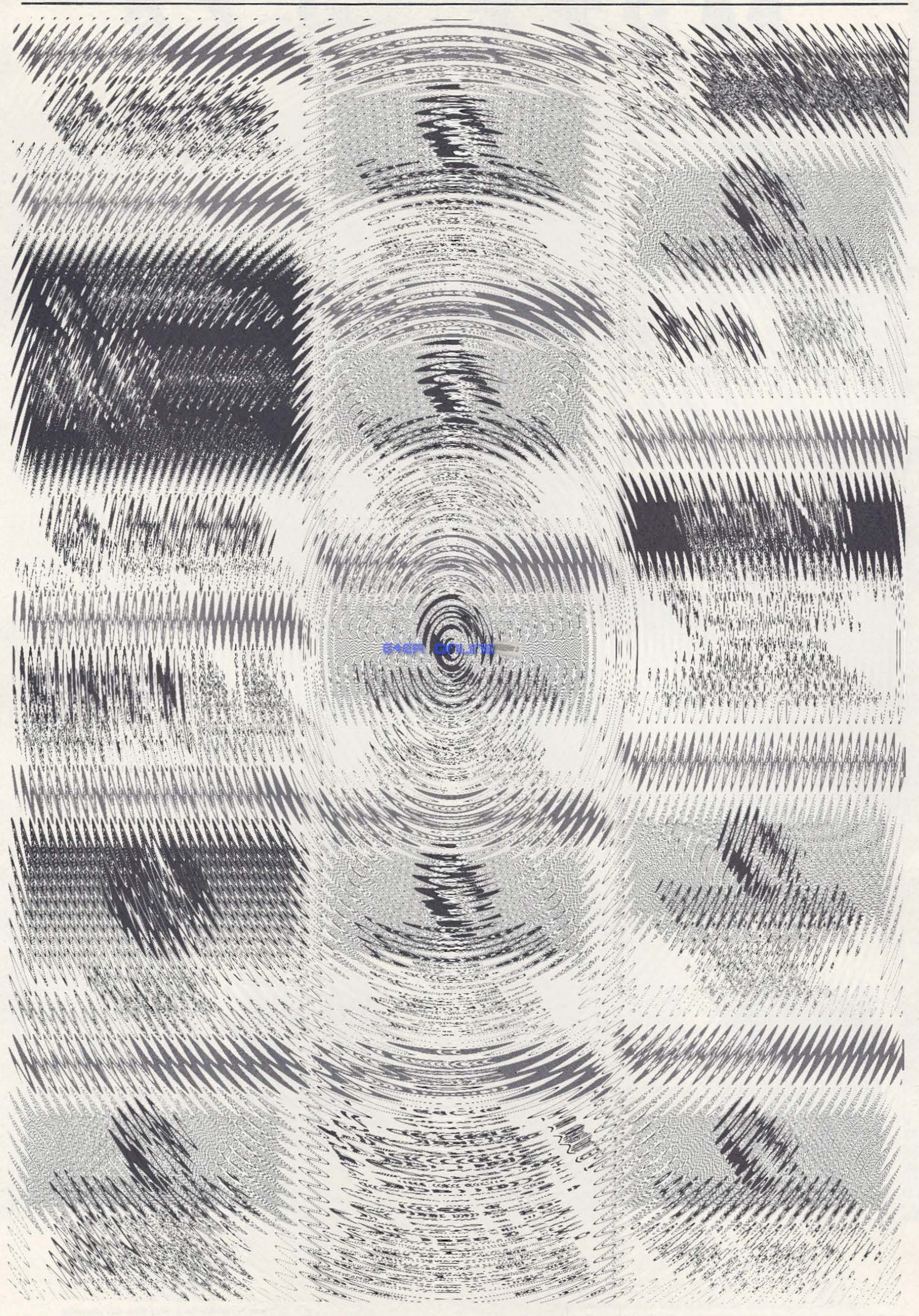
Listing 1. Maschinenroutinen zum Schach-Programm. Beachten Sie die Eingabehinweise auf Seite 53

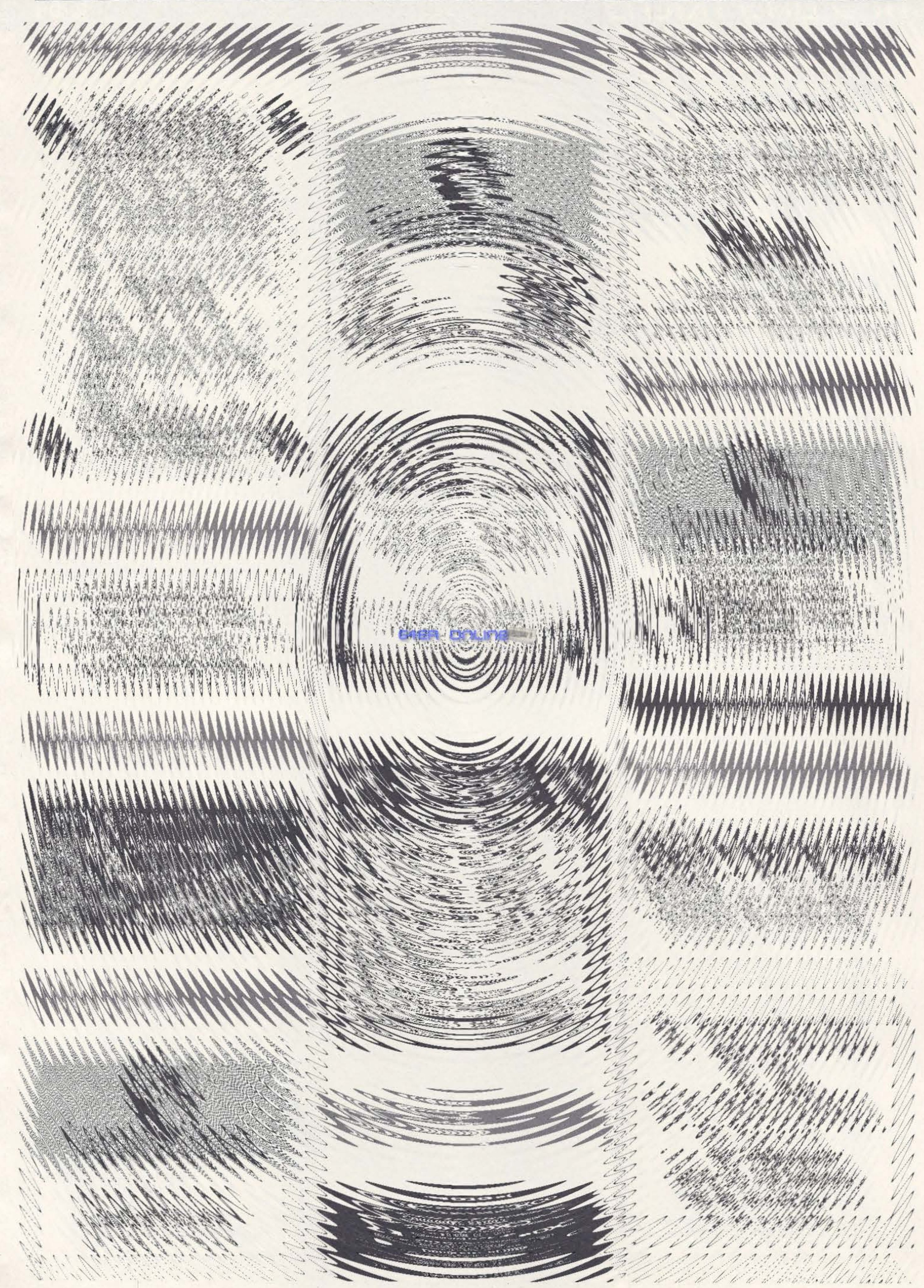
```
1 REM ****** SCHACH FUR DEN C-64 ******        <190>
2 REM **     BY BERND BETTERMANN      **       <181>
3 REM **        AMELITHER STR.21      **       <068>
4 REM **        3417 BODENFELDE       **       <134>
5 REM **  TEL:05572/330 AB 18'00      **       <060>
6 REM ********************************         <137>
7 :                                            <239>
10 IF PEEK(49408)<>162 THEN SYS 5517           <224>
20 GOTO 1000                                   <216>
70 :                                           <046>
80 REM ------ STELLUNGSBEWERTUNG ------        <093>
90 :                                           <066>
100 WZ=PEEK(ZF):POKE ZF,PEEK(X):POKE X,0       <086>
110 AA=USR(128):IF AA=128 THEN 520             <055>
120 :                                          <096>
200 AA=AA+WZ                                   <218>
210 IF PEEK(X-9)=1 OR PEEK(X-11)=1 THEN AA
    =AA+PEEK(ZF)-128                           <175>
220 IF PEEK(ZF-9)=1 OR PEEK(ZF-11)=1 THEN
    AA=AA-PEEK(ZF)-128                         <238>
400 :                                          <122>
500 IF AA>BW THEN BW=AA:BA=X:BZ=ZF:GOTO 52
    0                                          <141>
510 IF AA=BW AND RND(1)>.7 THEN BA=X:BZ=ZF     <231>
520 POKE X,PEEK(ZF):POKE ZF,WZ:RETURN          <228>
1000 :                                         <214>
1010 REM ----- VARIABLENDEFINITION ----        <180>
1020 :                                         <234>
1030 FOR X=49152 TO 49171:POKE X,[illegible]8:POKE
     X+100,128:NEXT X                          <079>
1040 FOR X=49172 TO 49251:POKE X,0:NEXT X      <068>
1050 FOR X=49172 TO 49242 STEP 10:POKE X,1
     28:POKE X+9,128:NEXT X                    <014>
1060 FOR X=49183 TO 49190:POKE X,1:POKE X+
     50,129:NEXT X                             <060>
1070 RESTORE:FOR X=49173 TO 49180:READ A:P
     OKE X,A:POKE X+70,A+128:NEXT X            <083>
1080 DATA 4,2,3,5,6,3,2,4                      <114>
1090 POKE 785,0:POKE 786,193                   <182>
2000 :                                         <198>
2010 REM ----- SPIELFELD AUSGEBEN -----        <107>
2020 :                                         <218>
2030 POKE 53280,0:POKE 53281,0:PRINT"{CLR,
     GREEN,2DOWN,2SPACE}@@@@@@@@"              <213>
2040 FOR Y=8 TO 1 STEP-1:PRINT CHR$(48+Y);
     "H";:FOR X=1 TO 8                         <076>
2050 W=PEEK(49162+Y*10+X)                      <036>
2060 IF W=0 THEN PRINT"@";:GOTO 2090           <243>
2070 IF W>128 THEN PRINT"{RVSON}";MID$("BS
     LTDK",W-128,1);"{RVOFF}";                 <095>
2080 IF W<128 THEN PRINT MID$("BSLTDK",W,1
     );                                        <213>
2090 NEXT X:PRINT                              <220>
2100 NEXT Y:PRINT"{2SPACE}TTTTTTTT":PRINT"
     {2SPACE}ABCDEFGH{2DOWN}":BW=-50           <088>
3000 :                                         <182>
3010 REM ------ ZUGEINGABE WEISS ------        <163>
3020 :                                         <202>
3030 INPUT"IHR ZUG ";Z$                        <222>
3040 IF LEN(Z$)<>4 THEN PRINT"{UP}":GOTO 3
     030                                       <244>
3050 IF MID$(Z$,1,1)<"A"OR MID$(Z$,1,1)>"H
     "THEN 3030                                <192>
3060 IF MID$(Z$,3,1)<"A"OR MID$(Z$,3,1)>"H
     "THEN 3030                                <076>
3070 IF MID$(Z$,2,1)<"1"OR MID$(Z$,2,1)>"8
     "THEN 3030                                <016>
3080 IF MID$(Z$,4,1)<"1"OR MID$(Z$,4,1)>"8
     "THEN 3030                                <159>
3090 AF=ASC(LEFT$(Z$,1))-64+10*VAL(MID$(Z$
     ,2,1))+49162                              <088>
3100 ZF=ASC(MID$(Z$,3,1))-64+10*VAL(RIGHT$
     (Z$,1))+49162                             <063>
3110 IF PEEK(AF)=0 OR PEEK(AF)>6 THEN 3030     <211>
3120 IF PEEK(ZF)>0 AND PEEK(ZF)<129 THEN 3
     030                                       <176>
3130 POKE ZF,PEEK(AF):POKE AF,0                <233>
4000 :                                         <166>
4010 REM --- SCHW. FIGURENPOS. HOLEN --        <206>
4020 :                                         <186>
4030 FOR X=49173 TO 49250:IF PEEK(X)>128 T
     HEN GOSUB 6000                            <205>
4040 NEXT X                                    <178>
5000 :                                         <150>
5010 REM ---- SCHW. ZUG AUSFUEHREN ----        <019>
5020 :                                         <170>
5030 POKE BZ,PEEK(BA):POKE BA,0:PRINT"{DOW
     N}WERT:";BW                               <081>
5040 PRINT"{DOWN}ZUG: ";                       <011>
5050 BB=BA-49172                               <217>
5055 IF BB>10 THEN BB=BB-10:GOTO 5055          <003>
5057 PRINT CHR$(BB+64);                        <085>
5060 PRINT CHR$(INT((BA-49172)/10)+49);"-"
     ;                                         <008>
5070 BB=BZ-49172                               <051>
5075 IF BB>10 THEN BB=BB-10:GOTO 5075          <087>
5077 PRINT CHR$(BB+64);                        <105>
5080 PRINT CHR$(INT((BZ-49172)/10)+49)         <094>
5085 IF BW=-50 THEN PRINT"{DOWN}ICH GEBE A
     UF":END                                   <085>
5090 FOR X=1 TO 1000:NEXT X:GOTO 2000          <091>
5100 :                                         <250>
5110 END                                       <030>
5120 :                                         <016>
6000 IF PEEK(X)<>129 THEN 6100                 <010>
6010 :                                         <144>
6020 REM --------- BAUERNZUEGE ---------       <026>
6030 :                                         <164>
6040 IF X>49233 AND PEEK(X-10)=0 AND PEEK(
     X-20)=0 THEN ZF=X-20:GOSUB 100            <037>
6050 IF PEEK(X-10)=0 THEN ZF=X-10:GOSUB 10
     0                                         <123>
6060 IF PEEK(X-9)>0 AND PEEK(X-9)<7 THEN Z
     F=X-9:GOSUB 100                           <093>
6070 IF PEEK(X-11)>0 AND PEEK(X-11)<7 THEN
      ZF=X-11:GOSUB 100                        <033>
6080 RETURN                                    <040>
6100 IF PEEK(X)<>130 THEN 6200                 <106>
6110 :                                         <244>
6120 REM ------- SPRINGERZUEGE --------        <116>
6130 :                                         <008>
6140 Z$="08121921":FOR Z=1 TO 7 STEP 2         <150>
6150 ZZ=VAL(MID$(Z$,Z,2))                      <003>
6160 IF PEEK(X+ZZ)<128 THEN ZF=X+ZZ:GOSUB
     100                                       <033>
6170 IF PEEK(X-ZZ)<128 THEN ZF=X-ZZ:GOSUB
     100                                       <115>
6180 NEXT Z:RETURN                             <164>
6200 IF PEEK(X)<>131 THEN 6400                 <089>
6210 :                                         <090>
6220 REM -------- LAEUFERZUEGE --------        <209>
6230 :                                         <110>
6240 FF=0:FOR Z=X+11 TO X+77 STEP 11:IF PE
     EK(Z)>6 THEN FF=1                         <074>
6250 IF FF=0 THEN ZF=Z:GOSUB 100:IF PEEK(Z
     )<>0 THEN FF=1                            <140>
6260 NEXT Z                                    <128>
6270 FF=0:FOR Z=X-11 TO X-77 STEP-11:IF PE
     EK(Z)>6 THEN FF=1                         <221>
6280 IF FF=0 THEN ZF=Z:GOSUB 100:IF PEEK(Z
     )<>0 THEN FF=1                            <170>
6290 NEXT Z                                    <158>
6300 FF=0:FOR Z=X+9 TO X+63 STEP+9:IF PEEK
     (Z)>6 THEN FF=1                           <125>
6310 IF FF=0 THEN ZF=Z:GOSUB 100:IF PEEK(Z
     )<>0 THEN FF=1                            <200>
6320 NEXT Z                                    <188>
6330 FF=0:FOR Z=X-9 TO X-63 STEP-9:IF PEEK
     (Z)>6 THEN FF=1                           <043>
6340 IF FF=0 THEN ZF=Z:GOSUB 100:IF PEEK(Z
     )<>0 THEN FF=1                            <230>
6350 NEXT Z                                    <218>
6360 RETURN                                    <066>
6400 IF PEEK(X)<>132 THEN 6600                 <171>
6410 :                                         <036>
6420 REM ---------- TURMZUEGE ----------       <165>
6430 :                                         <056>
6440 FF=0:FOR Z=X+10 TO X+70 STEP 10:IF PE
     EK(Z)>6 THEN FF=1                         <184>
6450 IF FF=0 THEN ZF=Z:GOSUB 100:IF PEEK(Z
     )<>0 THEN FF=1                            <086>
6460 NEXT Z                                    <074>
6470 FF=0:FOR Z=X-10 TO X-70 STEP-10:IF PE
     EK(Z)>6 THEN FF=1                         <043>
6480 IF FF=0 THEN ZF=Z:GOSUB 100:IF PEEK(Z
     )<>0 THEN FF=1                            <116>
6490 NEXT Z                                    <104>
6500 FF=0:FOR Z=X+1 TO X+7 STEP 1:IF PEEK(
     Z)>6 THEN FF=1                            <094>
6510 IF FF=0 THEN ZF=Z:GOSUB 100:IF PEEK(Z
     )<>0 THEN FF=1                            <146>
6520 NEXT Z                                    <134>
6530 FF=0:FOR Z=X-1 TO X-7 STEP-1:IF PEEK(
     Z)>6 THEN FF=1                            <166>
6540 IF FF=0 THEN ZF=Z:GOSUB 100:IF PEEK(Z
     )<>0 THEN FF=1                            <176>
6550 NEXT Z                                    <164>
6560 RETURN                                    <012>
6600 IF PEEK(X)<>5 THEN 6700                   <221>
6610 :                                         <236>
6620 REM ----------- DAMENZUEGE --------       <126>
6630 :                                         <000>
6640 GOSUB 6210:GOSUB 6410:RETURN              <186>
6700 :                                         <072>
6710 REM --------- KOENIGSZUEGE --------       <090>
6720 :                                         <092>
6730 Z$="01091011":FOR Z=1 TO 7 STEP 2         <027>
6740 ZZ=VAL(MID$(Z$,Z,2))                      <085>
6750 IF PEEK(X+ZZ)<7 THEN ZF=X+ZZ:GOSUB 10
     0                                         <236>
6760 IF PEEK(X-ZZ)<7 THEN ZF=X-ZZ:GOSUB 10
     0                                         <056>
6770 NEXT Z:RETURN                             <246>
6780 :                                         <152>
6790 REM ----------- ENDE ------------         <245>
6800 :                                         <172>

© 64'er
```

Listing 2. Schach — das Hauptprogramm







»Procedure« — oder der C 64 kann lernen

Strukturiertes Programmieren bleibt nicht länger ein Schlagwort. Definierbare Basic-Befehle kürzen manchen »Spaghetti-Code«.

Mit diesem Programm (Listing 1) können Sie eigene Basic-Befehle definieren, wie das auch in vielen anderen Programmiersprachen möglich ist. Die Befehlsangaben können dabei bis zu einer Zeile lang sein (80 Zeichen) und auch bereits definierte Befehle enthalten. Ein Beispiel dazu finden Sie in Listing 2. Dort wird in Zeile 20 und 30 eine Prozedur definiert, die aus Anfangskapital, Laufzeit und Zinssatz das Endkapital berechnet. In Zeile 70 wird die Prozedur mit »UMRECHNUNG« aufgerufen.

»Procedure« liegt im Speicherbereich von \$C000 bis \$C367 (Tabelle 1) und wird mit SYS 49514 initialisiert. Dabei wird der Vektor auf die Tabelle der Basic-Befehle von \$A7E1 auf \$C000 geändert. Die Adresse des Vektors steht in \$134 und \$135. Im ersten Teil von »Procedure« wird ständig geprüft, ob der Interpreter einen neu definierten Befehl gelesen hat. Wenn ja, sucht die Routine in einer eigenen Tabelle nach der Startadresse der gespeicherten Befehlsfolge. Handelt es sich um keinen zusätzlichen Befehl, macht der Computer weiter wie normal. Der Interpreter sucht in der Basic-Befehls-Tabelle des C 64 im ROM. »Procedure« versteht die Anweisungen in Tabelle 2.

Beachten Sie bitte, daß bei mehrfachen Definitionen eines Wortes immer die erste abgearbeitet wird, wenn Sie kein !NEW

\$F7/F8 Zähler
\$FA/FB nächster freier Platz in der Befehlsliste
\$FC/FD nächster freier Platz in der Befehlsfolgeliste
\$FE/FF verschiedene Funktionen
\$C000-C367 Programm
\$C370-CF9F Befehlsliste
\$CFA0-CFFD verschiedene Funktionen

Tabelle 1. Die Speicherbelegung von »Procedure«

!LIST zeigt alle neuen Befehle. Die Befehlsliste wird von oben nach unten durchgegangen und alle Befehle auf dem Bildschirm gedruckt.

!NEW löscht alle Eintragungen und setzt alle Zeiger auf den Anfangswert.

!DEF Befehlswort=:Befehlsfolge: – definiert neue Befehlsfolgen. Es wird eine Syntaxprüfung durchgeführt.

Tabelle 2. Mit diesen Befehlen können neue Basic-Befehlswörter kreiert werden

```
programm : procedure     c000 c370
-----------------------------------

c000 : 20 73 00 c9 21 f0 0c c9   70
c008 : 5f f0 13 20 79 00 4c 92   96
c010 : c0 ea ea 20 73 00 c9 96   90
c018 : f0 0b 4c eb c2 00 20 88   dc
c020 : c1 4c 33 c1 ea 20 60 c3   c5
c028 : 4c ae a7 ea a5 fe 85 7a   70
c030 : a5 ff 85 7b 60 a0 00 a9   04
c038 : 40 91 fa 20 7c c0 20 73   39
c040 : 00 c9 b2 f0 05 91 fa 4c   51
c048 : 3b c0 a9 40 91 fa 20 7c   c0
c050 : c0 a5 fc 91 fa 20 7c c0   78
c058 : a5 fd 91 fa 20 20 c1 ea   9f
c060 : 91 fa 20 5d c1 20 73 00   0d
c068 : c9 5f f0 08 91 fc 20 87   ae
c070 : c0 4c 65 c0 20 64 c1 20   34
c078 : cd c1 60 ea 18 e6 fa f0   22
c080 : 01 60 18 e6 fb 60 ea 18   33
c088 : e6 fc f0 01 60 18 e6 fd   a7
c090 : 60 ea 20 fd c1 ea 85 f7   a7
c098 : a9 cd 85 f8 a0 00 b1 f7   69
c0a0 : c9 40 f0 03 4c 04 c1 a2   57
c0a8 : 01 20 15 c1 b1 f7 c9 40   b9
c0b0 : f0 23 c5 ff d0 25 20 73   41
c0b8 : 00 c9 00 f0 69 c9 3a f0   6b
c0c0 : 65 85 ff e8 4c a9 c0 c8   ab
c0c8 : c8 b1 f7 85 7a c8 b1 f7   bc
c0d0 : 85 7b 4c ae a7 c6 f7 d0   2e
c0d8 : 02 c6 f8 20 1c c2 ea ea   d9
c0e0 : ea 20 15 c1 b1 f7 c9 40   da
c0e8 : d0 f7 a2 04 20 15 c1 ca   24
c0f0 : d0 fa b1 f7 c9 40 d0 19   bd
c0f8 : aa 20 79 00 85 ff 8a a2   d8
c100 : 01 4c a9 c0 20 79 00 4c   10
c108 : e7 a7 c6 7a d0 02 c6 7b   f3
c110 : 60 4c eb c1 ea 18 e6 f7   c4
c118 : f0 01 60 18 e6 f8 60 ea   31
c120 : 20 7c c0 a9 40 60 20 0d   85
c128 : c2 4c c7 c0 20 35 c0 20   09
c130 : 73 00 60 20 79 00 4c ae   e6
c138 : a7 00 20 79 00 ea 4c 92   c4
c140 : c0 a9 ea 85 80 85 81 85   85
c148 : 82 85 83 60 a9 c9 85 80   7a
c150 : a9 20 85 81 a9 f0 85 82   d8
c158 : a9 ef 85 83 60 20 7c c0   45
c160 : 20 41 c1 60 91 fc 20 4c   b7
c168 : c1 60 a9 00 8d 08 03 20   29
c170 : de c1 ea a9 c3 85 fd a9   d2
c178 : cd 85 fb a9 c0 8d 09 03   de
c180 : 60 20 15 c1 20 15 c1 60   e0
c188 : ae a0 cf ca bd a1 cf 85   07
c190 : 7b ca bd a1 cf 85 7a 8e   44
c198 : a0 cf 60 20 b8 c1 a5 7a   61
c1a0 : 9d a1 cf e8 a5 7b 9d a1   0f
c1a8 : cf e8 8e a0 cf 60 ea 85   5a
c1b0 : fa 85 fc 20 33 c3 60 ea   59
c1b8 : ae a0 cf e0 2a f0 01 60   b5
c1c0 : a2 00 8e a0 cf 4c 35 a4   97
c1c8 : a4 ff 85 fa 60 ea 20 87   19
c1d0 : c0 a6 02 e8 86 02 e0 2a   d1
c1d8 : f0 01 60 4c 35 a4 20 ae   41
c1e0 : c1 85 02 60 20 d1 c1 20   c8
c1e8 : 2c c1 60 ea ad fa cf 85   67
c1f0 : 7a ad fb cf 85 7b 20 79   e1
c1f8 : 00 4c e7 a7 ea 85 ff a5   33
c200 : 7a 8d fa cf a5 7b 8d fb   5e
c208 : cf a9 00 60 ea 48 8a 48   64
c210 : 98 48 20 9b c1 68 a8 68   1b
c218 : aa 68 60 ea ad fa cf 85   69
c220 : 7a ad fb cf 85 7b 60 a5   6b
c228 : 7a 8d fa cf a5 7b 8d fb   86
c230 : cf 20 73 00 c9 b2 f0 2c   3a
c238 : c9 00 f0 1c 4c 31 c2 20   5b
c240 : 73 00 c9 5f f0 07 c9 00   80
c248 : f0 0e 4c 3f c2 ad fa cf   5f
c250 : 85 7a ad fb cf 85 7b 60   d5
c258 : 4c 08 af ea ea 20 27 c2   c3
c260 : 20 2c c1 60 20 73 00 c9   44
c268 : 3a f0 d4 4c 58 c2 ea ea   f6
c270 : a9 00 85 f7 a9 cd 85 f8   8b
c278 : a0 00 b1 f7 c9 40 d0 65   30
c280 : 20 15 c1 b1 f7 c9 40 f0   82
c288 : 44 aa 38 e9 81 10 07 8a   36
c290 : 20 d2 ff 4c 80 c2 8a 38   5b
c298 : e9 80 aa a0 00 84 b5 e4   45
c2a0 : b5 f0 11 b9 9e a0 c8 38   cb
c2a8 : e9 80 10 03 4c a3 c2 e6   f1
c2b0 : b5 4c 9f c2 b9 9e a0 c8   70
c2b8 : 38 aa e9 80 10 07 8a 20   73
c2c0 : d2 ff 4c b4 c2 20 d2 ff   b4
c2c8 : a0 00 4c 80 c2 a9 20 20   c6
c2d0 : d2 ff a2 04 20 15 c1 ca   12
c2d8 : d0 fa 4c 78 c2 a9 11 20   45
c2e0 : d2 ff 4c 70 c2 a9 11 20   d1
c2e8 : d2 ff 60 c9 9b f0 07 c9   fc
c2f0 : a2 f0 4a 4c 92 c0 20 73   bd
c2f8 : 00 20 dd c2 4c ae a7 20   f1
c300 : 73 00 c9 00 f0 07 c9 3a   c9
c308 : f0 03 4c 08 af a0 00 a9   e1
c310 : 70 85 f7 a9 c3 85 f8 a9   15
c318 : 0c 85 f9 a9 ff 85 fa a9   06
c320 : ff 91 f7 20 15 c1 c6 fa   5a
c328 : d0 f7 c6 f9 d0 ed 20 6a   b6
c330 : c1 60 00 8d a0 cf a9 70   e3
c338 : 85 fc a9 00 60 20 ff c2   32
c340 : 4c ae a7 a5 fb c9 cf d0   71
c348 : 06 a5 fa c9 50 10 0e a5   22
c350 : fd c9 ce f0 01 60 a5 fc   a7
c358 : c9 b0 10 01 60 4c 35 a4   24
c360 : 20 43 c3 20 5d c2 60 ff   84
c368 : 8a 00 00 00 30 00 9a 00   60
```

Listing 1. »Procedure«. Das Listing muß mit dem MSE eingegeben werden. Vergessen Sie nach dem absoluten Laden nicht den NEW-Befehl.

eingegeben haben. Es können deshalb keine Befehlsfolgen umdefiniert werden.

Basic-Erweiterungen in der Praxis

Was geschieht mit den selbstdefinierten Befehlen? Das oben schon angerissene Thema ist relativ einfach zu beschreiben, vergleicht man das Programm mit einer Kartei. Es gibt eine Liste, in der alle Kunden (Wörter) eingetragen sind. In der Liste ist vermerkt, in welcher Schublade — stellen Sie sich einen großen Schrank vor — genaueres über den Kunden zu finden ist, oder in diesem Fall über die Befehlsfolge. Übrigens arbeiten fast alle Basic-Erweiterungen nach diesem Prinzip. Wenn nun »Procedure« auf !DEF stößt, wird das neue Befehlswort eingetragen, mit dem Vermerk in welcher Schublade die Befehlsfolge zu finden ist. Wird nun ein neu definiertes Befehlswort aufgerufen, wird die Nummer der Schublade in den Programmzeiger (Adressen $7A und $7B) geschrieben und die Programmausführung bei dieser Adresse fortgesetzt. Der Programmzeiger sagt nämlich dem C 64, was als nächstes bei welcher Adresse ausgeführt werden muß. Ist das Befehlswort der Pfeil (←), heißt das, daß die neue Befehlsfolge abgearbeitet ist und wieder ins Programm zurückgesprungen werden kann. Dazu wird in Adresse $7A und $7B einfach der aktuelle Wert geschrieben. Wie Sie sehen, ist es also sehr wichtig, daß am Ende der Befehlsdefinition der Pfeil (←) steht.

```
5 REM*****RECHEN-DEMO*****                          <063>
10 !NEW                                             <205>
15 :                                                <073>
20 !DEF UNTERPROGRAMM=:Q=1+P/100:KN=K*Q↑N:←         <120>
30 !DEF UMRECHNUNG=:UNTERPROGRAMM:PRINT"END
   KAPITAL NACH"N"JAHREN ="KN:←                     <119>
35 :                                                <093>
40 INPUT"ANFANGSKAPITAL{2SPACE}:";K                 <181>
50 INPUT"LAUFZEIT (JAHRE):";N                       <221>
60 INPUT"ZINS{12SPACE}:";P                          <014>
65 :                                                <123>
70 UMRECHNUNG                                       <066>
80 END                                              <208>
```

© 64'er

Listing 2. Demo zu »Procedure«

Nach einem Reset kann die Erweiterung mit SYS 49541 wieder aktiviert werden, wobei die neuen Befehle erhalten bleiben.

Tips zur Eingabe.

Das Maschinenprogramm (Listing 1) müssen Sie mit dem MSE eingeben und speichern. Nach dem Laden mit LOAD "PROCEDURE",8,1 (1,1) sollten Sie vor dem Initialisieren als erstes NEW eingeben, um bei weiteren Anweisungen keinen »OUT OF MEMORY ERROR« auszulösen. (Frank Pflüger/hm)

Hypra-Save

Hypra-Save ist eine Ergänzung zu Hypra-Load. Es speichert Programme 3- bis 5mal schneller und kann mit Hypra-Load verwendet werden.

Ein großer Nachteil der Diskettenstation VC 1541 ist die durch den seriellen Bus und durch das DOS V2.6 bedingte geringe Geschwindigkeit. Inzwischen gibt es mehrere Programme, die das Laden von Diskette beschleunigen. Mit der hier vorgestellten Routine geht jetzt auch das Speichern von Programmen mit dem C 64 wesentlich rascher.

Hypra-Save ist 3- bis 5mal so schnell wie die Originalroutine. Es verträgt sich mit Hypra-Load und vielen anderen, auch professionellen, Programmen und Basic-Erweiterungen. Zur Bedienung von Hypra-Save sollten Sie folgendes beachten: Die Eingabe muß mit dem MSE erfolgen. Nach dem Laden startet man es wie gewohnt mit RUN. Danach sollte man NEW eingeben, wenn man ein eigenes Programm schreiben will.

Hypra-Save kann Files mit oder ohne Verify speichern.

Gibt man vor dem Filenamen als erstes Zeichen einen Stern ein, so wird nicht verifiziert. Feststellbar an bis zu 5mal schnelleren Speicherzeiten. Mit Verify ist Hypra-Save etwa 3mal schneller als die Original-SAVE-Routine. Wer einen »25, WRITE ERROR« bisher nur aus der Literatur kennt, der kann getrost ohne Verify arbeiten. Selbstverständlich kann man weiterhin Programme überschreiben. Dann ist der Klammeraffe mit anzugeben. So überschreibt der Befehl SAVE "* @:name", 8 ein File, ohne die auf Diskette geschriebenen Blöcke zu prüfen, also ohne Verify. Hat der Computer alle Daten gesendet, wird im Gegensatz zur Original-SAVE-Routine nicht gewartet, bis das Laufwerk die Datei geschlossen hat. Dies macht sich besonders beim Überschreiben von Programmen bemerkbar. Die Floppystation arbeitet noch, während der Computer sich längst zurückgemeldet hat. Man darf die Diskette selbstverständlich nicht vor dem Erlöschen der roten LED aus dem Laufwerk nehmen.



Lebenslauf

Am 30.5.1968 wurde ich in München geboren. Nach einer fünfjährigen Zwischenstation in Erlangen zogen wir nach Bochum. Dort besuche ich zur Zeit die 11. Klasse des Gymnasiums Schillerschule.

Schon früh begann ich mich für Naturwissenschaften und Mathematik zu interessieren. So verwundert es nicht, wenn Ende 1983 ein C 64 gekauft werden mußte.

Vorerst behalf ich mir mit einem selbstgebauten Kassetteninterface, aber nach einem halben Jahr legte ich mir dann eine Diskettenstation VC 1541 zu. Deren relativ geringe Geschwindigkeit ist der Grund für Hypra-Save.

(Martin Pfost)

Beim Speichern von Programmen mit dem Klammeraffen kommt die 1541 häufig ins »Schleudern«, wie Sie vielleicht aus eigener Erfahrung wissen. So kann es passieren, daß einige Programme nicht mehr geladen werden können. Löschen Sie

```
programm : hypra-save    0801 0d3f
-----------------------------------

0801 : 0c 08 c1 07 9e 20 32 30  77
0809 : 36 32 00 00 00 78 a5 01  b5
0811 : 48 a9 34 85 01 a0 28 b9  15
0819 : 51 08 99 4f 01 88 d0 f7  46
0821 : a2 06 84 ac a9 d0 85 ad  0f
0829 : a9 a0 85 14 a9 08 85 15  21
0831 : b1 14 91 ac c8 d0 f9 e6  af
0839 : 15 e6 ad ca d0 f2 20 6d  86
0841 : 01 68 85 01 58 a2 da bd  79
0849 : a0 07 20 d2 ff e8 d0 f7  4a
0851 : 60 a5 ba c9 04 b0 03 4c  d6
0859 : ed f5 78 a6 01 a9 34 85  6d
0861 : 01 8e 9f d4 20 4f d4 ae  59
0869 : 9f d4 86 01 58 60 a9 50  04
0871 : 8d 32 03 a9 01 8d 33 03  5d
0879 : 60 0d 0d 09 0e 55 4c 54  2a
0881 : 52 41 53 41 56 45 20 36  ed
0889 : 34 20 41 4b 54 49 56 0d  8a
0891 : 49 4e 49 54 20 3a 20 53  d9
0899 : 59 53 20 33 36 35 0d ba  c1
08a1 : 8e a0 d4 ad 19 03 8d a1  8d
08a9 : d4 ad 18 03 8d a2 d4 a9  4f
08b1 : 9c 8d 18 03 a9 c1 8d 19  8b
08b9 : 03 ad 9f d4 85 01 ad 00  2c
08c1 : dd 8d 1f 01 58 a9 61 85  b0
08c9 : b9 a2 2c a0 00 b1 bb c9  03
08d1 : 2a d0 0a a2 4c c6 b7 e6  e2
08d9 : bb d0 02 e6 bc 8e a5 c3  b8
08e1 : a9 c1 48 a9 75 48 a5 b7  52
08e9 : d0 03 4c 10 f7 20 d5 f3  10
08f1 : 20 8f f6 a5 ba 20 0c ed  04
08f9 : a5 b9 20 b9 ed a9 00 85  f1
0901 : 90 20 dd ed 20 dd ed 20  bf
0909 : fe ed a5 90 f0 02 18 60  b9
0911 : a9 1a a2 c2 85 ac 86 ad  fb
0919 : a9 46 a2 01 85 14 86 15  eb
0921 : a2 05 a5 ba 20 0c ed a9  74
0929 : 6f 20 b9 ed a0 fd b9 15  df
0931 : c1 20 dd ed c8 d0 f7 a5  76
0939 : 14 20 dd ed a5 15 20 dd  d2
0941 : ed a9 1e 20 dd ed a0 00  5e
0949 : b1 ac 20 dd ed c8 c0 1e  78
0951 : 90 f6 20 fe ed a5 ac 18  33
0959 : 69 1e 85 ac 90 02 e6 ad  d8
0961 : a5 14 18 69 1e 85 14 90  c3
0969 : 02 e6 15 ca d0 b4 a5 ba  3c
0971 : 20 0c ed a9 6f 20 b9 ed  03
0979 : a0 fb b9 1a c1 20 dd ed  39
0981 : c8 d0 f7 20 fe ed 78 a2  3a
0989 : 00 8e 00 dd ad 11 d0 29  85
0991 : ef 8d 11 d0 bd ae c2 9d  3d
0999 : 00 c5 e8 d0 f7 ad 00 dd  79
09a1 : 30 fb 20 fe c1 a2 00 bd  63
09a9 : ae c3 9d 00 c5 e8 d0 f7  77
09b1 : 20 fe c1 20 8e fb a2 01  1a
09b9 : a5 ac 9d 01 c5 e8 a5 ad  d1
09c1 : 9d 01 c5 e8 20 d1 fc b0  53
09c9 : 38 a5 ad c9 c0 90 1d c9  11
09d1 : c6 b0 19 69 16 85 ad a9  fa
09d9 : 34 85 01 b1 ac 48 ad 9f  49
09e1 : d4 85 01 a5 ad e9 15 85  f6
09e9 : ad 68 b0 02 b1 ac 9d 01  30
09f1 : c5 e8 20 db fc e0 ff 90  a6
09f9 : cb 20 d1 fc b0 03 a9 ff  b2
0a01 : 2c a9 00 48 8d 00 c5 8e  18
0a09 : 01 c5 20 fe c1 a2 01 68  db
0a11 : d0 b2 18 a9 00 48 ad 11  91
0a19 : d0 09 10 8d 11 d0 ad 1f  b0
0a21 : 01 8d 00 dd 78 a9 34 85  55
0a29 : 01 ad a2 d4 8d 18 03 ad  45
0a31 : a1 d4 8d 19 03 68 ae a0  32
0a39 : d4 9a 60 38 ad 1f 01 09  63
0a41 : 10 8d 1f 01 b0 cd b9 00  60
0a49 : c5 85 95 a2 00 2c 00 dd  a8
0a51 : 50 fb 2c 00 dd 10 05 e8  ee
0a59 : d0 f8 f0 b6 a9 10 8d 00  0a
0a61 : dd aa 66 95 6a 66 95 6a  e5
0a69 : 4a 4a 8d 00 dd 8a 66 95  33
0a71 : 6a 66 95 6a 4a 4a 8d 00  ee
0a79 : dd 8a 66 95 6a 66 95 6a  ed
0a81 : 4a 4a 8d 00 dd 8a 66 95  4b
0a89 : 6a 66 95 6a 4a 4a 8d 00  06
0a91 : dd ea ea ea a9 00 8d 00  cc
0a99 : dd c8 d0 aa 60 a0 00 98  a0
0aa1 : 59 00 c5 c8 d0 fa 85 14  a8
0aa9 : 20 a7 c1 88 a5 14 4c aa  a0
0ab1 : c1 4d 2d 57 4d 2d 45 b7  12
0ab9 : 01 a0 00 84 11 a9 02 8d  1c
0ac1 : 00 18 a9 04 2c 00 18 f0  bd
0ac9 : fb a9 00 8d 00 18 a2 03  9c
0ad1 : ca d0 fd a2 0a ad 00 18  15
0ad9 : 4a 6a 4a 66 85 0a 0a 66  55
0ae1 : 85 ad 00 18 4a 6a 4a 66  2e
0ae9 : 85 0a 0a 66 85 ad 00 18  b9
0af1 : 4a 6a 4a 66 85 0a 0a 66  6d
0af9 : 85 ad 00 18 8e 00 18 4a  36
0b01 : 6a 4a 66 85 0a 0a 66 85  70
0b09 : a5 85 91 30 45 11 85 11  f0
0b11 : c8 d0 aa 88 60 85 31 20  34
0b19 : 46 01 b1 30 48 20 4a 01  03
0b21 : 68 91 30 a5 11 f0 ed 4c  fb
0b29 : 43 e8 78 a9 0a 8d 00 18  71
0b31 : a2 00 88 d0 fd ca d0 fa  7f
0b39 : a9 0a 85 69 a9 00 85 30  87
0b41 : a9 03 20 a2 01 a9 04 20  76
0b49 : a2 01 4c 3d 04 a5 00 a2  d9
0b51 : 01 86 00 29 02 f0 10 a6  f0
0b59 : 98 86 32 20 2e 04 20 af  a8
0b61 : 03 20 03 04 4c 69 f9 a2  f3
0b69 : 08 20 35 03 a2 0a 20 35  94
0b71 : 03 a2 08 20 f5 03 a2 0a  e2
0b79 : 20 f5 03 a5 8c 30 e8 4c  90
0b81 : 69 f9 86 32 86 98 a5 8c  ac
0b89 : 10 61 b5 83 d0 5d bd 31  79
0b91 : 04 20 a2 01 c8 a6 32 a9  4c
0b99 : 80 95 83 a5 80 95 00 a5  79
0ba1 : 81 95 01 b1 30 f0 1c 20  9f
0ba9 : 21 f1 a6 82 f6 b5 d0 02  21
0bb1 : f6 bb a0 00 a5 80 91 30  b2
0bb9 : c8 a5 81 91 30 a5 80 c5  a4
0bc1 : 22 f0 02 84 8c a6 32 a9  87
0bc9 : 00 85 30 85 33 85 2e 85  6c
0bd1 : 36 85 0c 85 50 a9 bb 85  ca
0bd9 : 34 bd 31 04 85 2f 20 e9  df
0be1 : f5 85 3a bd 32 04 20 a3  ea
0be9 : f7 a6 32 b5 83 f0 52 20  c0
0bf1 : 2e 04 ad 00 1c 29 10 d0  7a
0bf9 : 03 4c 81 f5 20 10 f5 a2  e1
0c01 : 09 50 fe b8 ca d0 fa a9  7b
0c09 : ff 8d 03 1c ad 0c 1c 29  11
0c11 : 1f 09 c0 8d 0c 1c a9 ff  df
0c19 : a2 05 8d 01 1c b8 50 fe  88
0c21 : b8 ca d0 fa a0 bb b1 0c  98
0c29 : 50 fe b8 8d 01 1c c8 d0  8e
0c31 : f5 b1 30 50 fe b8 8d 01  03
0c39 : 1c c8 d0 f5 50 fe 4c 00  da
0c41 : fe 60 86 32 2c 24 04 86  58
0c49 : 98 b5 83 f0 f4 20 2e 04  cc
0c51 : 20 0a f5 a0 bb b1 0c 50  22
0c59 : fe b8 4d 01 1c d0 19 c8  65
0c61 : d0 f3 b1 30 50 fe b8 4d  18
0c69 : 01 1c d0 0c c8 c0 fd d0  5a
0c71 : f1 a6 32 a9 00 95 83 60  f3
0c79 : 4c c5 f6 bd 31 04 85 31  c9
0c81 : bd 32 04 85 0d 60 05 01  f3
0c89 : 06 04 58 20 19 f1 a9 84  7c
0c91 : d5 a7 f0 05 95 a7 20 42  b2
0c99 : d0 a9 40 8d f9 02 a9 01  58
0ca1 : 85 83 20 07 d1 90 03 4c  17
0ca9 : f8 cf 20 3e de f6 b5 a9  28
0cb1 : 00 85 8b 85 8d a9 80 85  3b
0cb9 : 8c a5 80 85 06 a9 e0 85  25
0cc1 : 00 a5 00 30 fc f0 24 c9  15
0cc9 : 01 f0 ea a5 18 85 06 a5  c3
0cd1 : 19 85 07 a2 00 a9 b0 20  13
0cd9 : 7d d5 20 99 d5 a9 e2 20  f3
0ce1 : 7d d5 20 99 d5 a5 8c d0  e2
0ce9 : cc f0 ce 4c 23 db a0 c0  80
0cf1 : a2 d6 20 6a d4 a0 d0 a2  2f
0cf9 : c0 20 6a d4 20 00 c0 48  94
0d01 : a0 d6 a2 c0 20 6a d4 68  46
0d09 : 60 ad fa ff 48 ad fb ff  e0
0d11 : 48 a9 fb 8d fa ff a9 d3  dc
0d19 : 8d fb ff 84 ad 86 15 a0  d9
0d21 : 00 84 ac 84 14 a2 06 b1  f1
0d29 : ac 91 14 c8 d0 f9 e6 ad  90
0d31 : e6 15 ca d0 f2 68 8d fb  0f
0d39 : ff 68 8d fa ff 60 b2 4c  95
```

MSE-Listing zu Hypra-Save. Beachten Sie bitte die Eingabehinweise auf Seite 53.

daher ein Programm erst mit dem SCRATCH-Befehl und speichern Sie erst dann die neue Version (ohne »@«) ab.

Anstelle von STOP dient bei Hypra-Save 64 die RESTORE-Taste zum Abbrechen. Allerdings wird die Programmdatei dann nicht geschlossen und erscheint im Directory mit einem Stern. Möchte man die nicht geschlossene Datei löschen, so geht das nicht mit dem SCRATCH-, sondern nur mit dem VALIDATE-Befehl. Übrigens blinkt die rote LED nach dem Drücken von RESTORE. Liest man dann den Fehlerkanal aus, so erhält man die Meldung »51, OVERFLOW IN RECORD«, gefolgt von Track und Sektor des letzten Blocks. Diese Fehlermeldung weist sonst auf einen Übertragungsfehler hin, der von zu vielen Geräten am seriellen Bus herrühren kann.

Sollte man eine merkwürdige Fehlernummer wie zum Beispiel 61 oder 71 erhalten, hilft meist nur ein Aus- und Einschalten der Floppy. Nach STOP/RESTORE oder nach einem Reset ist der SAVE-Vektor zurückgesetzt. Hypra-Save läßt sich dann mit SYS 365 wieder aktivieren.

Das Programm verschiebt sich nach dem Starten. Der eine Teil von Hypra-Save liegt ab $0150 bis $0177, der andere von $D000 bis $D5FF im Speicher. Wird die SAVE-Routine aufgerufen, so wird über den geänderten SAVE-Vektor in die Initialisierungs-Routine nach $0150 gesprungen. Dort wird geprüft, ob die anzusprechende Gerätenummer größer als 3 ist. Wenn ja, blendet die Routine das RAM unter den I/O-Bausteinen ein. Dann kopiert es den Speicherbereich von $C000 bis $C5FF, der nach dem Speichern wieder zurückgeschrieben wird, nach $D600 bis $DBFF (RAM) und verschiebt die eigentlichen SAVE-Routinen von $D000 bis $D5FF (ROM) nach $C000 bis $C5FF. Denn die I/O-Bausteine können nicht vom darunterliegenden RAM angesprochen werden.

Durch diese Speicherbelegungstechnik wird kein normalerweise sichtbares RAM verändert und nur ein minimaler Bereich im Stack von der Hypra-Save-Routine belegt. Vom Programmteil bei $C000 wird das File zuerst wie gewohnt durch die normalen IEC-Busroutinen geöffnet. Dann testet Hypra-Save ob kein Fehler aufgetreten ist und die Daten gesendet werden können. Ist kein Fehler aufgetreten, so wird die SAVE-Routine für das Diskettenlaufwerk in das Floppy-RAM geschrieben und gestartet. Dieser Programmteil liegt von $0146 bis $01D9 im Floppy-RAM. Er enthält hauptsächlich die Transferroutine, mit der sofort nach dem Aufrufen das Diskettenstationsprogramm übertragen, in den Bereich von $0300 bis $04A0 geschrieben und dann gestartet wird.

Der wesentlich zur Geschwindigkeitserhöhung beitragende Datentransfer beruht auf der gleichzeitigen Übertragung von je zwei Bit und auf dem erheblich eingeschränkten Handshakebetrieb, mit dem beide Geräte ihre Bereitschaft zum Transfer signalisieren.

Die hohe SAVE-Geschwindigkeit wird aber auch vom Schreibalgorithmus erzielt, der die für das eventuell durchzuführende Verify benötigte Zeit erheblich verkürzt. Hierzu ein Beispiel, um die Arbeitsweise dieses Algorithmus zu verdeutlichen:

Wenn sich der Schreib-/Lesekopf in einer halben Diskettenumdrehung über dem Header des gewünschten Blocks befindet, die zu schreibenden Daten im Puffer stehen und codiert sind, so hat die Originalroutine den Pufferinhalt nach einer halben Umdrehung geschrieben und nach einer weiteren ganzen Umdrehung verifiziert. Diese Methode benötigt 1,5 Umdrehungen pro Block, also drei für zwei Blöcke.

Hypra-Save geht anders vor. Es verwendet zwei Puffer. Nach einer halben Umdrehung wird der erste, nach einer weiteren halben der zweite Block geschrieben. Es wird dann noch eine Umdrehung gebraucht, um beide Sektoren zu verifizieren. Man benötigt also nur zwei Rotationen für zwei Blöcke, der Zeitgewinn beträgt etwa 0,1 Sekunden pro Block. Sind sämtliche Bytes übertragen, so stellt der Computer wie erwähnt das RAM von $C000 bis $C5FF wieder her, springt in die Digitalisierungsroutine, schaltet die ursprüngliche Speicherbelegung ein und kehrt ins aufrufende Programm zurück. Das Laufwerk schließt währenddessen das File, wofür es mehr Zeit brauchen kann, wenn beim Überschreiben das alte Programm gelöscht werden muß.

(Martin Pfost/hm)

Der Bitmap-Compander

Mit diesem kurzen Programm können Grafikbildschirme, die auf Diskette abgespeichert werden, auf die Hälfte komprimiert werden.

Jeder, der schon einmal eine hochauflösende Grafik auf Diskette abgespeichert hat, weiß, daß sie 33 Blocks benötigt. Das entspricht den 8 KByte des hochauflösenden Grafikbildschirms. Es besteht aber die Möglichkeit, diese Grafikbildschirme je nach Inhalt auf 10 bis 20 Blocks zu komprimieren. Das Verfahren beruht darauf, häufig wiederkehrende Sequenzen durch kürzere zu ersetzen. Dazu kann man sich die Bitmap als eine lange Kette von Nullen und Einsen vorstellen. Diese Kette wird jetzt in 4-Bit-Blöcke aufgeteilt. Solche Blöcke bezeichnet man als Halb-Byte oder Nibbles. Es gibt 16 verschiedene solcher Nibbles, wovon einige häufiger und andere seltener vorkommen. Jedem Nibble wird nun ein neuer Code zugeordnet, wobei das häufigste Nibble »0000«, den kürzesten Code »0« bekommt. Alle anderen 15 Codes müssen mit einer »1« beginnen. Einen Code »00« darf es nicht mehr geben, denn er könnte bei der Decodierung nicht mehr von dem Code »0« unterschieden werden. Es läßt sich nicht umgehen, viele Codes länger als 4 Bit zu machen. Die komplette Code-Tabelle ist in Tabelle 1 zu sehen. Doch die langen Codes kommen viel seltener vor als die kurzen, so daß bei einem »normalen« Bild viele Bits gespart werden können. Theoretisch wäre es möglich, daß eine komprimierte Bitmap länger wird als das Original. Aber selbst die Fotos aus der »Diashow« lassen sich komprimieren. Sir Winston Churchill magert zum Beispiel auf 24 Blocks ab. Bilder, wie sie üblicherweise in Gafik-Adventures vorkommen, lassen sich meist auf unter 15 Blocks zusammendrücken.

Tippen Sie das Programm (Listing 1) mit dem MSE ab und speichern Sie es, damit es jederzeit mit LOAD "BMC.EXE",8,1 geladen werden kann (auf der Leserservice-Diskette unter "BIT-MAPCOMPANDER" gespeichert). Der Aufruf des Companders funktioniert vom Basic aus mit einem SYS-Befehl. Die Parameter werden wie beim OPEN-Befehl angehängt.

Laden einer komprimierten Bitmap:
SYS 52798,Filenummer,Geräteadresse,Kanalnummer,"Name"

Speichern einer komprimierten Bitmap:
SYS 52736,Filenummer,Geräteadresse,Kanalnummer,"Name,P,W"

Mit der Filenummer wird die Anfangsadresse der Bitmap ausgewählt.

Filenummer	Bitmap-Adresse	
1	$2000 = 8192	(sichtbares Bild bei Hi-Eddi)
2	$4000 = 16384	
3	$6000 = 24576	
5	$A000 = 40960	(Bitmap bei Hires-3)
7	$E000 = 57344	(Bitmap bei Simons Basic)

Die Bitmaps 0, 4 und 6 sind theoretisch möglich, aber nicht sinnvoll, da der Videocontroller sie nicht adressieren kann. Die Gerätenummer ist 8 für das Floppy-Laufwerk und die Kanalnummer eine beliebige Zahl von 2 bis 14.

Beispiele für Aufrufe:

Speichern einer Simons Basic-Bitmap:
SYS 52736,7,8,2,"Name,P,W"

Laden dieser Bitmap in Hires-3:
SYS 52798,5,8,2,"Name"

Das Programm belegt den Bereich von $CE00 bis $CF19 (52736 bis 53017). Es belegt somit keinen Basic-Speicherplatz und verträgt sich auch mit Hires-3 und Simons Basic, solange der Befehl MEM nicht verwendet wird.

(Hans Haberl/ah)

```
 programm : bmc.exe          ce00 cf19
-----------------------------------------
ce00 : 20 89 ce a6 b8 20 c9 ff   21
ce08 : a9 08 85 fc 78 a9 34 85   67
ce10 : 01 a0 00 b1 f8 a2 37 86   26
ce18 : 01 58 48 4a 4a 4a 4a 20   01
ce20 : 9d ce 68 29 0f 20 9d ce   6a
ce28 : e6 f8 d0 02 e6 f9 a5 f9   c7
ce30 : 29 1f c9 1f d0 d6 a5 f8   8b
ce38 : c9 40 d0 d0 f0 42 20 89   24
ce40 : ce a6 b8 20 c6 ff 20 cf   20
ce48 : ff 85 fa 20 cf ff 85 fb   d7
ce50 : a9 08 85 fc 20 bc ce 0a   35
ce58 : 0a 0a 0a 48 20 bc ce 68   e7
ce60 : 05 fd a0 00 78 a2 34 86   06
ce68 : 01 91 f8 a2 37 86 01 58   21
ce70 : e6 f8 d0 02 e6 f9 a5 90   3d
ce78 : f0 da a5 f8 c9 40 d0 d4   e9
ce80 : 20 cc ff a5 b8 20 c3 ff   56
ce88 : 60 20 fd ae 20 be e1 a5   18
ce90 : b8 0a 0a 0a 0a 0a 85 f9   0c
ce98 : a9 00 85 f8 60 a8 b9 09   06
cea0 : cf aa b9 e9 ce a4 fc 0a   8a
cea8 : 26 fa 88 d0 09 48 a5 fa   e7
ceb0 : 20 d2 ff a0 08 68 ca d0   de
ceb8 : ee 84 fc 60 a2 0f a5 fa   63
cec0 : 3d f9 ce dd e9 ce f0 03   48
cec8 : ca d0 f3 86 fd bd 09 cf   59
ced0 : aa a4 fc 06 fb 26 fa 88   ba
ced8 : d0 07 20 cf ff 85 fb a0   8b
cee0 : 08 ca d0 ef 84 fc a5 fd   42
cee8 : 60 00 a0 a8 b0 b8 c0 c8   eb
cef0 : f0 d0 d8 e0 f4 e8 f8 fc   0f
cef8 : 80 80 f8 f8 f8 f8 f8 f8   43
cf00 : fc f8 f8 f8 fc f8 fc fc   5b
cf08 : e0 01 05 05 05 05 05 05   e1
cf10 : 06 05 05 05 06 05 06 06   28
cf18 : 03 ff fc 68 ff ff ff 68   38
```

Listing 1. zum Programm »Bitmap-Compander«. Bitte mit dem MSE eingeben.

Nibbles	Codes	Hex	Länge
0000	0	$00	1
0001	10100	$A0	5
0010	10101	$A8	5
0011	10110	$B0	5
0100	10111	$B8	5
0101	11000	$C0	5
0110	11001	$C8	5
0111	111100	$F0	6
1000	11010	$D0	5
1001	11011	$D8	5
1010	11100	$E0	5
1011	111101	$F4	6
1100	11101	$E8	5
1101	111110	$F8	6
1110	111111	$FC	6
1111	100	$80	3

Tabelle 1. Tabelle aller möglichen Codes

Wie spät ist es bitte?

Haben Sie auch schon beim Programmieren jedes Zeitgefühl verloren und vielleicht einen Termin verpaßt? Mit diesem Uhr-Programm dürfte das nicht mehr passieren. Zusätzlich zur ständigen Zeitanzeige kann noch eine Alarmfunktion aufgerufen werden.

Mit »Piep Piep Piep« erinnert Sie dieses Programm daran, daß es schon wieder nach Mitternacht ist. Es ist vielleicht besser, die neueste Version zu speichern und ins Bett zu gehen. Aber genug der Vorrede, gehen wir »in medias res«.

Das Programm CIA-Uhr wird bei jedem Interrupt des C 64 einmal abgearbeitet, also 60mal pro Sekunde. Dabei wird die Echtzeituhr des C 64 abgefragt, die Alarmzeit überprüft und die Uhrzeit auf den Bildschirm gedruckt. Im Gegensatz zu TI oder TI$ hat die Echtzeituhr eine große Ganggenauigkeit. Die Variable TI wird nämlich nur bei jedem Interrupt hochgezählt. Ein Interrupt muß aber nicht jede 60stel Sekunde stattfinden. Beispielsweise werden beim Speichern oder Laden von Programmen wesentlich weniger Interrupts pro Sekunde vom Computer ausgelöst als im READY-Modus.

Was bedeutet Interrupt?

Der C 64 unterbricht etwa jede 60stel Sekunde das laufende Programm und überprüft, ob eine Taste (vielleicht RUN/STOP-RESTORE) gedrückt wurde oder ob ein angeschlossenes Gerät Daten empfangen oder senden kann.

Wird ein Interrupt ausgelöst, sieht der C 64 in zwei Adressen ($314, $315) der Zeropage nach, bei welcher Adresse das Interrupt-Programm beginnt. Im Normalfall bei $EA31. Da die Zeropage zum RAM-Speicher gehört, kann der C 64 auf eigene Interrupt-Routinen umgelenkt werden. Im Normalfall endet ein eigenes Interrupt-Programm mit dem Assemblerbefehl JMP $EA31, was heißt, daß der Computer mit der »serienmäßigen« Routine weitermachen soll.

Die CIAs 6526 (Complex Interface Adapter, vielseitiger Ein-/Ausgabe-Baustein) regeln beim C 64 alles was mit Ein- und Ausgaben zusammenhängt. Dieser Baustein hat zwei Timer eingebaut. Einer davon steuert in der CIA auch eine 24-Stunden-Uhr (AM/PM) mit einer Auflösung von 1/10 Sekunde. Die Uhr wird mit TOD (Time Of Day) bezeichnet. Die Zeit steht im BCD-Format in den Registern 8, 9, 10 und 11 der CIA 1. Die Ganggenauigkeit des Timer ergibt sich aus der Tatsache, daß er mit Netzfrequenz geregelt wird.

Der Trick mit dem Interrupt

»CIA-Uhr« steht im Speicher von $C000 (49152) bis $C230 (49712). Das Programm wird mit SYS 49152, "Uhrzeit im HHMMSS-Format", "Alarmzeit" initialisiert. Dabei wird der Interrupt-Vektor (Inhalt der Adressen $314, $315) des C 64 so verändert, daß er auf $C036 zeigt. Danach wird die angegebene Uhrzeit und Alarmzeit gesetzt. Pro Interrupt wird nun die Uhrzeit rechts oben am Bildschirm gedruckt und mit der Alarmzeit verglichen. Stimmen die Zeiten überein, fängt der Computer an zu piepen. Und zwar so lange, bis der Alarm mit SYS 49704 abgeschaltet wird.

Das Programm unterscheidet zwischen Vor- und Nachmittag, obwohl die Uhrzeit nur im 12-Stunden-Format ausgegeben wird. Setzt man beispielsweise die Uhrzeit auf 1 Uhr und die Alarmzeit auf 13 Uhr, erscheint in der Anzeige 01:00:00;0. Der Alarm wird aber erst 12 Stunden später ausgelöst. Die »0« hinter dem Strichpunkt gibt die 10tel Sekunden an.

Das Programm »CIA-Uhr« kann mit allen Basic-Erweiterungen benutzt werden, die den Interruptvektor nicht verändern und nicht den Speicherbereich von Adresse $C000 bis $C023 belegen.

Die Anzeige der Uhr erfolgt rechts oben am Bildschirm. Mit POKE 49311,0 kann sie auf die linke Seite verlagert werden.

Tips zur Eingabe und Benutzung

Das Programm muß mit dem MSE eingegeben werden. Nach dem Laden mit LOAD"UHR",8,1 (1,1) muß NEW eingegeben werden, damit kein OUT OF MEMORY ERROR auftritt. Bei RUN/STOP-RESTORE verschwindet die Anzeige vom Bildschirm, während die Uhr intern aber weiterläuft. Mit SYS 49152 wird die Anzeige wieder aktiviert. Das Maschinen-Programm kann auch in andere Speicherbereiche verschoben werden. Man sollte dazu wissen, daß in den Bereichen $C0A3 bis $C0A6 und $C137 bis $C14D Tabellen stehen.

(Jörg Dorchain/hm)

```
programm : uhr               c000 c230
----------------------------------
c000 : 78 a9 36 8d 14 03 a9 c0   0e
c008 : 8d 15 03 58 ad 0f dc 29   05
c010 : 7f 8d 0f dc a9 85 8d 0d   cc
c018 : dc ad 0e dc 09 80 8d 0e   d1
c020 : dc 4c eb c1 ad 0d dc 29   5e
c028 : 04 f0 03 20 bb c0 ad 43   68
c030 : c1 f0 03 20 c1 c0 a0 00   d3
c038 : ad 0b dc ae 08 dc 29 10   a4
c040 : 18 4a 4a 4a 4a 20 9b c0   ef
c048 : ad 0b dc ae 08 dc 29 0f   b2
c050 : a2 00 20 9b c0 bd a3 c0   78
c058 : 20 9b c0 e8 ad 0a dc 29   84
c060 : f0 4a 4a 4a 4a 20 9b c0   e7
c068 : ad 0a dc 29 0f 20 9b c0   58
c070 : bd a3 c0 20 9b c0 e8 ad   f2
c078 : 09 dc 29 f0 4a 4a 4a 4a   0c
c080 : 20 9b c0 ad 09 dc 29 0f   8e
c088 : 20 9b c0 bd a3 c0 20 9b   56
c090 : c0 e8 ad 08 dc 20 9b c0   ef
c098 : 4c 31 ea 18 69 b0 99 1e   f9
c0a0 : 04 c8 60 0a fe 0b a9 20   91
c0a8 : 8d 04 d4 a9 07 8d 05 d4   3c
c0b0 : a9 f4 8d 06 d4 a9 0f 8d   e9
c0b8 : 18 d4 60 a9 09 8d 43 c1   15
c0c0 : 60 6c 37 c1 a9 14 8d 00   ce
c0c8 : d4 a9 1a 8d 01 d4 a9 21   49
c0d0 : 8d 04 d4 a0 14 4c 10 c1   10
c0d8 : a9 20 8d 04 d4 a0 0a 4c   88
c0e0 : 10 c1 a9 cf 8d 00 d4 a9   b5
c0e8 : 22 8d 01 d4 a9 21 8d 04   8e
c0f0 : d4 a0 14 4c 10 c1 a9 20   99
c0f8 : 8d 04 d4 a0 0a 4c 10 c1   97
c100 : a0 00 8c 41 c1 a9 c4 8d   83
c108 : 37 c1 a9 c0 8d 38 c1 60   05
c110 : 8c 42 c1 a9 1d 8d 37 c1   01
c118 : a9 c1 8d 38 c1 ce 42 c1   2b
c120 : d0 14 ac 41 c1 b9 39 c1   a0
c128 : 8d 37 c1 c8 b9 39 c1 8d   62
c130 : 38 c1 c8 8c 41 c1 60 c4   3a
c138 : c0 d8 c0 e2 c0 f3 c0 00   9f
c140 : c1 00 00 00 10 00 00 00   03
c148 : 00 00 00 00 00 00 20 a6   16
c150 : c0 20 06 e2 20 fd ae 20   eb
c158 : 9e ad 20 8f ad 20 a3 b6   9f
c160 : 86 14 84 15 c9 06 f0 0a   59
c168 : a9 00 aa a8 18 d8 b8 4c   95
c170 : 48 b2 a0 ff c8 b1 14 38   14
c178 : e9 30 90 ec 99 45 c1 c0   87
c180 : 05 d0 f1 a0 ff c8 b1 14   b3
c188 : c9 3a b0 dc c0 05 d0 f5   99
c190 : a0 ff a2 00 c8 b9 45 c1   cb
c198 : 0a 0a 0a 0a 8d 44 c1 c8   ff
c1a0 : b9 45 c1 0d 44 c1 9d 4b   6d
c1a8 : c1 e8 e0 03 d0 e6 a2 00   45
c1b0 : a0 0b f8 bd 4b c1 f0 b0   b3
c1b8 : c9 24 d0 03 a9 00 18 b0   84
c1c0 : a7 c9 12 90 04 e9 12 09   cc
c1c8 : 80 99 00 dc e8 88 bd 4b   11
c1d0 : c1 c9 60 b0 93 99 00 dc   64
c1d8 : e8 88 bd 4b c1 c9 60 b0   2a
c1e0 : 87 99 00 dc d8 a9 00 8d   c5
c1e8 : 08 dc 60 20 4e c1 20 0c   06
c1f0 : c2 ad 0f dc 09 80 8d 0f   d1
c1f8 : dc 20 51 c1 20 0c c2 78   cf
c200 : a9 24 8d 14 03 a9 c0 8d   3d
c208 : 15 03 58 60 a0 ff c8 b9   62
c210 : 45 c1 f0 fa c0 09 d0 03   6f
c218 : 4c 0b e2 a0 ff a9 00 c8   95
c220 : 99 45 c1 c0 08 d0 f8 60   90
c228 : a9 00 8d 43 c1 4c a6 c0   38
```

Listing zur CIA-Uhr. Es muß mit dem MSE (siehe Seite 53) eingegeben werden.

Disketten-Monitor

Neben den hervorragenden Möglichkeiten Disketteninhalte zu bearbeiten, läßt sich der Bildschirminhalt nach oben und unten scrollen. Außerdem stehen DOS-Erweiterungen, ähnlich dem DOS 5.1, zur Verfügung, die das Diskettenhandling auch vom Basic her wesentlich erleichtern.

Der Disketten-Monitor wird mit SYS 49152 gestartet und steht im Bereich von $C000 bis $CC71. Intern wird noch ein Puffer von $CF00 bis $CFFF benötigt, auf den der Benutzer Zugriff hat. Der Puffer kann in den Speicher geschrieben und auch wieder gelesen werden, so daß die Möglichkeit besteht, ihn mit einem Maschinensprachemonitor zu bearbeiten. Es werden keine Veränderungen am DOS der 1541 vorgenommen. Soweit bekannt ist, ist der Disketten-Monitor mit allen Basic-Programmen und Hypra-Load kompatibel. Zu beachten ist lediglich, daß der Monitor den Interrupt-Vektor und das DOS den Basic-Vektor benutzen.

Die Befehle können entweder mit Parameter oder ohne Parameter eingegeben werden. Sind Parameter nicht erforderlich, werden sie im folgenden in Klammern angegeben (im Monitor keine Klammern eingeben). Ansonsten muß zwischen den Parametern ein Leerzeichen stehen. Die Angaben in eckigen Klammern sollen lediglich eine Eselsbrücke zum besseren Verständnis der Befehle darstellen. Alle Eingaben werden mit »RETURN« abgeschlossen. Nach Diskettenzugriffen werden automatisch eine oder mehrere (Fehler-) Meldungen des Floppy-Laufwerkes ausgegeben. (Horst Reichart/ah)

Schreib-/Lesebefehle des Disketten-Monitors

Befehl	Beschreibung
R (Track Sektor) [Read Block]	liest den angegebenen Block in den Disketten-Puffer (liest den Block, auf den zuletzt zugegriffen wurde).
W (Track Sektor) [Write Block]	schreibt den Puffer in den angegebenen Block. (schreibt den Puffer in den letzten Block.)
N [Next Block]	liest den logisch nächsten Block in den Puffer. Nach dem letzten Block wird End of File! ausgegeben.
+	liest den physikalisch nächsten Block in den Puffer.
—	liest den physikalisch vorhergehenden Block in den Puffer.
S [Show]	zeigt aktuelle Track- und Sektornummern an.

Puffer bearbeiten

Befehl	Beschreibung
M (von bis) [Memory]	gibt den gesamten Pufferinhalt aus. (gibt den Pufferinhalt von bis aus. Es können Adressen von $00 bis $FF angegeben werden.) Der Puffer kann durch Überschreiben geändert werden.
CRSR Up	der Puffer kann mit den Cursortasten gescrollt werden.
CRSR Down	Steht noch eine Adresse des Dumps auf dem Bildschirm wird bei der nächsten (vorhergehenden) Adresse der Dump fortgesetzt. Ansonsten wird der Puffer von Anfang (Ende) ausgegeben.
P Adresse [Put Buffer to Memory]	der Puffer wird in den Speicher geschrieben. Dazu muß eine vierstellige Adresse angegeben werden. Es darf auch unter das Basic-ROM ($A000 bis $BFFF) und unter das Kernal-ROM ($E000 bis $FFFF) geschrieben werden, da dort der Puffer am wenigsten stört. Nicht zulässig ist der I/O-Bereich ($D000 bis $DFFF) und natürlich der Bereich $C000 bis $CFFF. Es werden jedoch keine Fehlermeldungen ausgegeben.
G Adresse [Get Buffer from Memory]	der Puffer wird aus dem angegebenen Bereich gelesen. Es gilt das gleiche wie beim P-Befehl.
C [Copy]	druckt den Pufferinhalt auf einem MPS 801 aus.

Die Diskbefehle

Befehl	Beschreibung
@ (Kommando)	liest den Fehlerkanal. (sendet das angegebene Kommando zum Floppy-Laufwerk. Das Leerzeichen nach @ muß entfallen.)
$ oder @$	listet das Directory. Es werden jeweils 20 Files gelistet. Dann stoppt der Ausdruck. Er kann nun mit RUN/STOP abgebrochen oder mit einer beliebigen Taste fortgesetzt werden.
B [BAM]	zeigt die BAM an. Alle Angaben werden in Hex ausgegeben, zum Beispiel Track 18 entspricht $12.
A Track Sektor [Allocate]	kennzeichnet den angegebenen Block in der BAM als belegt. Es wird anschließend zur Kontrolle der B-Befehl aufgerufen.
A T Track [Allocate Track]	kennzeichnet den angegebenen Track als belegt.
A A [Allocate All]	kennzeichnet die gesamte Diskette als belegt.
F Track Sektor [Free]	gibt den angegebenen Block frei.
F T Track [Free Track]	gibt den angegebenen Track frei.
F A [Free All]	gibt die gesamte Diskette frei.
T Track [Tracking]	zeigt die ersten 8 Bytes eines jeden Sektors des angegebenen Tracks an.
X [Exit]	Disketten-Monitor verlassen

Unbenutzte Befehle

Befehl	Beschreibung
U	führt einen Kaltstart des Monitors aus (kann später für Erweiterungen benutzt werden).
H	es gilt das Gleiche wie unter U.

Das DOS

Nach dem Verlassen des Monitors werden die Befehle des DOS in der linken oberen Ecke des Bildschirms ausgegeben.

Befehl	Beschreibung
DLOAD	lädt ein Basic-Programm. Es kann direkt aus dem Directory ohne nachfolgenden Doppelpunkt geladen werden. Soll ein File absolut geladen werden, ist dies mit LOAD "Name",8,1 weiterhin möglich.
DSAVE	speichert ein Basic-Programm.
DVERIFY	vergleicht ein Programm mit dem Speicherinhalt.
DIR	listet das Directory ohne Programmverlust. Es gilt das bei »$« gesagte. Anders als zum Beispiel in Simons Basic sind keine weiteren Eingaben zulässig.
DISK	sendet einen Befehl zum Floppy-Laufwerk. Der Befehl muß in Gänsefüßchen stehen.
DERROR	liest den Fehlerkanal.
DMON	zurück zum Disketten-Monitor.

Die Befehle des Disketten-Monitors



```
programm : disk-monitor      c000 cc75
--------------------------------------

c000 : a9 80 8d 8a 02 78 a9 6a  fd
c008 : 8d 14 03 a9 c0 8d 15 03  68
c010 : a9 00 8d 12 d0 ad 11 d0  bf
c018 : 29 7f 8d 11 d0 a9 81 8d  02
c020 : 1a d0 58 a9 00 8d 71 c9  b3
c028 : a9 01 8d 21 d0 a2 d8 a0  a0
c030 : c9 20 3a c9 a2 0d a9 2e  67
c038 : 20 e4 c7 a9 00 8d 6d c9  a7
c040 : 20 cf ff c9 2e f0 f9 c9  67
c048 : 20 f0 f5 a2 13 dd 73 c9  33
c050 : d0 12 8e 6c c9 a9 c0 48  d8
c058 : a9 33 48 bd 87 c9 48 bd  c8
c060 : 9b c9 48 60 ca 10 e6 4c  5f
c068 : 34 c0 ad 19 d0 8d 19 d0  0a
c070 : 10 30 ad 12 d0 c9 64 b0  94
c078 : 0c a9 00 8d 20 d0 a9 64  03
c080 : 8d 12 d0 d0 1a c9 d2 b0  01
c088 : 0c a9 02 8d 20 d0 a9 d2  70
c090 : 8d 12 d0 d0 0a a9 00 8d  7e
c098 : 12 d0 a9 07 8d 20 d0 4c  13
c0a0 : bc fe a9 c0 48 a9 e6 48  5c
c0a8 : 08 48 48 48 ad 0d dc 58  57
c0b0 : 4c 31 ea ad 71 c9 d0 11  d0
c0b8 : 78 a9 6a 8d 14 03 a9 c0  d3
c0c0 : 8d 15 03 a9 81 8d 1a d0  5c
c0c8 : 58 60 ad 71 c9 d0 fa 78  ea
c0d0 : a9 31 8d 14 03 a9 ea 8d  3c
c0d8 : 15 03 a9 01 8d 20 d0 a9  6a
c0e0 : 00 8d 1a d0 58 60 58 a5  7d
c0e8 : c6 d0 03 4c 81 ea ad 77  76
c0f0 : 02 c9 11 d0 78 a5 d6 c9  d9
c0f8 : 18 d0 f0 a5 cf f0 06 a9  59
c100 : 00 85 c6 f0 e6 a5 d1 85  80
c108 : 5c a5 d2 85 5d a9 19 85  2f
c110 : 5e a0 01 20 e0 c1 c9 3a  ba
c118 : f0 1d c6 5e d0 0c a9 00  28
c120 : 85 5f 85 c6 20 0a c2 4c  85
c128 : 4c c1 38 a5 5c e9 28 85  d8
c130 : 5c b0 de c6 5d d0 da 20  7d
c138 : ea c1 b0 2e a9 00 85 c6  33
c140 : a5 5f 69 08 90 06 20 5f  69
c148 : c8 4c 5f c1 a8 20 72 c8  2d
c150 : 98 20 d5 c7 20 5c c8 a9  c2
c158 : 08 20 43 c8 20 ea c8 a2  1c
c160 : 18 a0 01 20 0c e5 a9 05  ad
c168 : 85 cd 4c 81 ea c9 91 d0  fc
c170 : f9 a5 d6 d0 f5 a5 cf f0  b9
c178 : 06 a9 00 85 c6 f0 eb a5  f2
c180 : d1 85 5c a5 d2 85 5d a9  02
c188 : 19 85 5e a0 01 20 e0 c1  28
c190 : c9 3a f0 1f c6 5e d0 0e  55
c198 : a9 00 85 c6 20 20 c2 20  ca
c1a0 : 0a c2 a9 f8 d0 20 18 a5  4e
c1a8 : 5c 69 28 85 5c 90 dc e6  ff
c1b0 : 5d d0 d8 20 ea c1 b0 b2  94
c1b8 : a9 00 85 c6 20 20 c2 38  1a
c1c0 : a5 5f e9 08 90 13 a8 20  15
c1c8 : 75 c8 98 20 d5 c7 20 5c  a0
c1d0 : c8 a9 08 20 43 c8 20 ea  44
c1d8 : c8 a2 00 a0 01 4c 63 c1  89
c1e0 : b1 5c c8 c9 20 b0 02 09  cc
c1e8 : 40 60 c0 28 d0 02 38 60  4c
c1f0 : 20 e0 c1 c9 20 f0 f3 20  c3
c1f8 : 21 c8 0a 0a 0a 0a 85 5f  07
c200 : 20 e0 c1 20 21 c8 05 5f  30
c208 : 85 5f a9 ff 85 cc a5 cf  9c
c210 : f0 0a a5 ce a4 d3 91 d1  1b
c218 : a9 00 85 cf a5 5f 18 60  93
c220 : a6 d2 20 27 c2 a6 f4 e8  23
c228 : e8 e8 86 ad 86 61 a2 00  da
c230 : 86 ac a9 28 85 60 a0 c0  db
c238 : a2 03 38 b1 ac 91 60 98  be
c240 : d0 f9 c6 ad c6 61 ca 10  b6
c248 : f1 a9 20 a6 d2 86 61 84  db
c250 : 60 a0 27 91 60 88 10 fb  7f
c258 : a2 00 a0 00 20 0c e5 60  dd
c260 : 20 ca c0 a9 80 8d 71 c9  18
c268 : a9 0e 8d 20 d0 a9 06 8d  0d
c270 : 21 d0 a0 00 b9 83 ca f0  e6
c278 : 06 20 d2 ff c8 d0 f5 20  6e
c280 : 87 c2 68 68 4c 7b e3 a9  13
c288 : 92 8d 08 03 a9 c2 8d 09  3c
c290 : 03 60 20 73 00 c9 44 f0  7b
c298 : 06 20 79 00 4c e7 a7 a2  f5
c2a0 : 00 8e 72 c9 a0 01 bd 60  87
c2a8 : ca f0 08 d1 7a d0 15 e8  7b
c2b0 : c8 d0 f3 20 fb a8 ad 72  82
c2b8 : c9 0a aa bd 76 ca 48 bd  43
c2c0 : 75 ca 48 60 ee 72 c9 ad  bd
c2c8 : 72 c9 c9 07 f0 0b bd 60  91
c2d0 : ca f0 03 e8 d0 f8 e8 d0  0a
c2d8 : cb a0 00 f0 bc a9 01 2c  87
c2e0 : a9 00 85 0a 20 f6 c2 20  31
c2e8 : 6f e1 4c ae a7 20 f6 c2  0e
c2f0 : 20 59 e1 4c ae a7 a9 00  8d
c2f8 : 20 bd ff a2 08 a0 00 20  11
c300 : ba ff 20 06 e2 20 57 e2  d5
c308 : 60 20 00 e2 8a a8 a2 08  5d
c310 : 4c ba ff 20 75 c4 4c ae  c9
c318 : a7 4c f9 e0 a9 0f 20 c3  9b
c320 : ff 20 2f c3 20 c0 ff b0  dd
c328 : f0 20 c3 ff 4c ae a7 a9  45
c330 : 00 20 bd ff a9 0f a8 a2  ab
c338 : 08 20 ba ff 20 06 e2 20  fd
c340 : 57 e2 60 20 e9 c4 4c ae  78
c348 : a7 20 cf ff c9 0d f0 13  e2
c350 : 20 ed c7 90 3c 8d 50 ca  72
c358 : 20 39 c8 20 ed c7 90 31  0d
c360 : 8d 53 ca 20 6c c3 20 59  66
c368 : c8 4c 75 c4 20 7c c8 ad  b1
c370 : 6c c9 c9 01 f0 1c a9 31  4c
c378 : 20 b0 c8 a2 0d 20 c6 ff  64
c380 : a2 00 20 cf ff 9d 00 cf  b1
c388 : e8 d0 f7 20 cc ff 20 a3  6f
c390 : c8 60 20 25 c9 a2 0d 20  5b
c398 : c9 ff a2 00 bd 00 cf 20  65
c3a0 : d2 ff e8 d0 f7 20 cc ff  7a
c3a8 : a9 32 20 b0 c8 4c 8e c3  39
c3b0 : ae 00 cf f0 0c ad 01 cf  42
c3b8 : 8e 50 ca 8d 53 ca 4c 63  56
c3c0 : c3 a2 c5 a0 ca 4c 3a c9  e5
c3c8 : ae 50 ca ad 53 ca 18 69  c5
c3d0 : 01 dd af c9 90 e2 a9 00  ac
c3d8 : e8 e0 24 d0 db a2 01 d0  cc
c3e0 : d7 ae 50 ca ad 53 ca 38  8d
c3e8 : e9 01 b0 cc ca d0 02 a2  98
c3f0 : 23 bc af c9 88 98 d0 c0  a8
c3f8 : 20 cf ff c9 0d f0 5a 20  3b
c400 : ed c7 90 69 f0 67 c9 24  dc
c408 : 48 20 ca c0 a9 80 8d 71  e3
c410 : c9 68 8d 50 ca a9 00 8d  90
c418 : 53 ca a2 d5 a0 ca 20 3a  89
c420 : c9 ad 50 ca 20 d5 c7 20  3d
c428 : 5f c8 20 5f c8 ad 53 ca  bc
c430 : 20 d5 c7 20 59 c8 20 4b  24
c438 : c9 a9 08 a0 00 20 43 c8  8c
c440 : 20 ea c8 20 5f c8 ae 50  a3
c448 : ca ac 53 ca c8 98 dd af  bf
c450 : c9 b0 06 8d 53 ca 4c 2d  bc
c458 : c4 a9 00 8d 00 cf 8d 01  59
c460 : cf 8d 50 ca 8d 53 ca 8d  1d
c468 : 71 c9 20 b3 c0 60 20 cf  6b
c470 : ff c9 0d d0 4a 20 5f c8  66
c478 : a9 00 85 90 20 ca c0 a9  43
c480 : 08 85 ba 20 b4 ff a9 6f  ce
c488 : 85 b9 20 96 ff 20 a5 ff  5c
c490 : c9 30 d0 20 20 a5 ff c9  6c
c498 : 30 d0 08 a9 91 20 d2 ff  cd
c4a0 : 4c b9 c4 48 a9 30 20 d2  45
c4a8 : ff 68 4c b4 c4 20 a5 ff  69
c4b0 : 24 90 70 05 20 d2 ff d0  13
c4b8 : f4 20 ab ff 4c b3 c0 c9  a0
c4c0 : 24 d0 03 4c e9 c4 48 20  bd
c4c8 : ca c0 a9 08 85 ba 20 b1  70
c4d0 : ff a9 6f 85 b9 20 93 ff  1b
c4d8 : 68 20 a8 ff 20 cf ff c9  8e
c4e0 : 0d d0 f6 20 ae ff 4c 75  1e
c4e8 : c4 20 5f c8 20 ca c0 a9  5c
c4f0 : 14 85 fd a9 00 85 90 a9  3d
c4f8 : 3e 85 bb a9 ca 85 bc a9  3c
c500 : 01 85 b7 a9 08 85 ba a9  d2
c508 : 60 85 b9 20 d5 f3 20 7a  10
c510 : c5 a4 90 d0 58 a0 06 84  11
c518 : fb 20 a5 ff a6 fc 85 fc  ef
c520 : a4 90 d0 49 a4 fb 88 d0  57
c528 : ee a4 fc 20 cd bd a9 20  5d
c530 : 20 d2 ff 20 a5 ff a6 90  d3
c538 : d0 33 aa f0 06 20 d2 ff  17
c540 : 4c 33 c5 20 5f c8 c6 fd  ef
c548 : d0 1f 20 ab ff a9 14 85  ce
c550 : fd 20 b3 c0 a5 c5 c9 40  92
c558 : f0 fa 20 ed f6 f0 0e 20  fa
c560 : 7a c5 a4 90 d0 07 20 ca  53
c568 : c0 a0 04 d0 aa 20 42 f6  36
c570 : 20 b3 c0 a9 00 85 c6 4c  af
c578 : 78 c4 a5 ba 20 b4 ff a5  06
c580 : b9 20 96 ff 60 20 ed c7  3d
c588 : 90 13 a8 a9 08 85 97 20  4c
c590 : 39 c8 20 39 c8 20 19 c9  e2
c598 : d0 f8 20 ea c8 60 a0 00  5c
c5a0 : 8c 6f c9 88 8c 70 c9 20  1b
c5a8 : cf ff c9 0d f0 15 20 ed  9f
c5b0 : c7 90 10 8d 6f c9 20 cf  da
c5b8 : ff c9 0d f0 06 20 ed c7  a6
c5c0 : 8d 70 c9 ac 6f c9 20 64  1c
c5c8 : c8 20 72 c8 98 20 d5 c7  c7
c5d0 : 20 5c c8 a9 08 20 43 c8  a6
c5d8 : 20 ea c8 4c c6 c5 20 cf  e4
c5e0 : ff c9 0d f0 22 c9 20 f0  f8
c5e8 : f5 c9 54 f0 6d c9 41 d0  c1
c5f0 : 03 4c 9f c6 20 02 c8 90  30
c5f8 : 60 8d 5b ca 20 39 c8 20  7e
c600 : ed c7 90 55 8d 5e ca a9  ea
c608 : 03 cd 6c c9 d0 03 a9 41  94
c610 : 2c a9 46 8d 57 ca 20 ca  36
c618 : c0 20 7c c8 ad 5b ca 48  92
c620 : 20 2c c8 8e 5b ca 8d 5c  55
c628 : ca ad 5e ca 48 20 2c c8  82
c630 : 8e 5e ca 8d 5f ca a2 0f  47
c638 : 20 c9 ff a2 00 bd 55 ca  6a
c640 : 20 d2 ff e8 e0 0b d0 f5  7c
c648 : 68 8d 5e ca 68 8d 5b ca  5e
c650 : 20 cc ff 20 a3 c8 20 66  a8
c658 : c3 60 20 39 c8 20 ed c7  4f
c660 : c9 00 f0 f5 c9 24 b0 f1  88
c668 : 85 5f 20 c1 c6 a9 03 cd  3e
c670 : 6c c9 d0 03 18 90 01 38  d0
c678 : 08 a5 5f 48 20 ce c6 68  98
c680 : 68 a2 01 8e 6c c9 20 63  a8
c688 : c3 a2 01 86 c6 a2 0d 8e  80
c690 : 77 02 a9 49 20 c6 c4 a2  2c
c698 : 00 8e 71 c9 4c f6 ca 20  5d
c6a0 : c1 c6 a9 23 85 fb a9 03  78
c6a8 : cd 6c c9 d0 03 18 90 01  6d
c6b0 : 38 08 a5 fb 48 20 ce c6  23
c6b8 : 68 c6 fb d0 f5 48 4c 7f  6e
c6c0 : c6 20 ca c0 a2 12 a9 00  c2
c6c8 : 8e 71 c9 4c b8 c3 ba e8  71
c6d0 : e8 9a 68 aa bc af c9 0a  f9
c6d8 : 0a aa 28 b0 04 a9 00 f0  c7
c6e0 : 19 98 9d 00 cf a9 ff 9d  32
c6e8 : 01 cf 9d 02 cf 98 38 e9  ef
c6f0 : 11 a8 b9 d3 c9 9d 03 cf  73
c6f8 : d0 0f 9d 00 cf 9d 01 cf  45
c700 : 9d 02 cf 9d 03 cf f0 01  ba
c708 : 68 ba ca ca ca ca 9a 60  07
c710 : 20 cf ff c9 0d f0 22 20  72
c718 : 3a c7 20 ca c0 78 a9 35  78
c720 : 85 01 b1 fb 99 00 cf c8  7c
c728 : d0 f8 a9 37 85 01 20 b3  0e
c730 : c0 a9 00 8d 50 ca 8d 53  af
c738 : ca 60 20 39 c8 20 02 c8  89
c740 : 85 fc 20 39 c8 20 02 c8  9a
c748 : 85 fb a0 00 60 20 cf ff  39
c750 : c9 0d f0 e5 20 3a c7 b9  ff
c758 : 00 cf 91 fb c8 d0 f8 4c  b3
c760 : 31 c7 a9 04 aa a0 00 20  50
c768 : ba ff a9 00 20 bd ff 20  bc
c770 : c0 ff a2 04 a2 04 20 c9  b7
c778 : ff 4c ca c0 20 5f c8 20  c8
c780 : 5f c8 a2 e0 a0 ca 20 3a  5d
c788 : c9 ad 50 ca 20 d5 c7 a2  aa
c790 : eb a0 ca 20 3a c9 ad 53  d1
c798 : ca 20 d5 c7 4c 5f c8 20  04
c7a0 : cf ff c9 0d f0 09 c9 24  4a
c7a8 : f0 29 c9 42 f0 26 60 20  e9
c7b0 : c0 c7 20 5f c8 20 cc ff  09
c7b8 : a9 04 20 c3 ff 4c b3 c0  96
c7c0 : a0 ff 8c 70 c9 c8 8c 6f  85
c7c8 : c9 20 62 c7 20 7c c7 20  78
c7d0 : c3 c5 60 60 60 48 4a 4a  a0
c7d8 : 4a 4a 20 17 c8 aa 68 29  08
c7e0 : 0f 20 17 c8 48 8a 20 d2  dd
c7e8 : ff 68 4c d2 ff a9 00 8d  f1
c7f0 : 6e c9 20 39 c8 c9 20 d0  6f
c7f8 : 09 20 39 c8 c9 20 d0 0f  78
c800 : 18 60 20 21 c8 0a 0a 0a  8e
c808 : 0a 8d 6e c9 20 39 c8 20  dd
c810 : 21 c8 0d 6e c9 38 60 18  b7
c818 : 69 f6 90 02 69 06 69 3a  42
c820 : 60 c9 3a 08 29 0f 28 90  c1
c828 : 02 69 08 60 a2 30 38 e9  4d
c830 : 0a 90 03 e8 b0 f9 69 3a  55
c838 : 60 20 cf ff c9 0d d0 02  e8
c840 : 68 68 60 85 97 20 5c c8  23
c848 : b9 00 cf 20 d5 c7 c8 d0  5a
c850 : 03 ee 6d c9 c6 97 d0 ed  a7
c858 : 60 20 5c c8 a9 20 2c a9  98
c860 : 0d 4c d2 ff ad 6d c9 d0  57
c868 : 06 cc 70 c9 b0 01 60 68  8f
c870 : 68 60 20 5f c8 a9 3a a2  04
c878 : 2e 4c e4 c7 20 ca c0 a9  ad
c880 : 0f a8 a2 08 20 ba ff a9  b8
c888 : 00 20 bd ff 20 c0 ff a9  63
c890 : 0d a8 a2 08 20 ba ff a9  c6
c898 : 01 a2 3f a0 ca 20 bd ff  73
c8a0 : 4c c0 ff a9 0d 20 c3 ff  62
c8a8 : a9 0f 20 c3 ff 4c b3 c0  0c
c8b0 : 8d 49 ca ad 50 ca 48 20  07
```

```
c8b8 : 2c c8 8e 50 ca 8d 51 ca  ea
c8c0 : ad 53 ca 48 20 2c c8 8e  76
c8c8 : 53 ca 8d 54 ca a2 0f 20  ac
c8d0 : c9 ff a2 00 bd 48 ca 20  cb
c8d8 : d2 ff e8 e0 0d d0 f5 68  00
c8e0 : 8d 53 ca 68 8d 50 ca 4c  f6
c8e8 : cc ff 98 38 e9 08 a8 20  a3
c8f0 : 5c c8 a9 12 20 d2 ff a2  3b
c8f8 : 08 b9 00 cf 29 7f c9 20  cd
c900 : b0 04 a9 2e d0 03 b9 00  ee
c908 : cf 20 d2 ff a9 00 85 d4  f6
c910 : c8 ca d0 e5 a9 92 4c d2  34
c918 : ff 20 ed c7 90 03 99 00  23
c920 : cf c8 c6 97 60 a2 0f 20  8f
c928 : c9 ff a2 00 bd 40 ca 20  e3
c930 : d2 ff e8 e0 08 d0 f5 4c  cf
c938 : cc ff 86 fb 84 fc a0 00  d8
c940 : b1 fb f0 06 20 d2 ff c8  16
c948 : d0 f6 60 20 7c c8 a9 31  c7
c950 : 20 b0 c8 a2 0d 20 c6 ff  3c
c958 : a2 00 20 cf ff 9d 00 cf  89
c960 : e8 e0 08 d0 f5 20 cc ff  68
c968 : 4c a3 c8 ff 06 00 db ff  88
c970 : 00 00 cb 3a 57 52 41 46  44
c978 : 4e 40 24 4d 47 50 53 43  64
c980 : 58 42 55 48 2b 2d 54 c5  51
c988 : c3 c3 c5 c5 c3 c4 c4 c5  58
c990 : c7 c7 c7 c7 c2 ca bf bf  27
c998 : c3 c3 c3 84 48 48 dd dd  b8
c9a0 : af 6d e8 9d 0f 4c 7b 9e  72
c9a8 : 5f f5 ff ff c7 e0 f7 00  65
c9b0 : 15 15 15 15 15 15 15 15  b0
c9b8 : 15 15 15 15 15 15 15 15  b8
c9c0 : 15 13 13 13 13 13 13 13  c2
c9c8 : 12 12 12 12 12 12 11 11  c2
c9d0 : 11 11 11 01 03 07 1f 1f  f2
c9d8 : 93 1f 20 12 20 20 20 20  09
c9e0 : 20 20 20 20 31 35 34 31  0c
c9e8 : 20 44 49 53 4b 2d 4d 4f  d9
c9f0 : 4e 49 54 4f 52 20 56 20  a2
c9f8 : 32 2e 35 20 20 20 20 20  57
ca00 : 20 20 0d 0d 20 20 20 20  d9
ca08 : 20 20 20 20 20 28 43 29  e7
ca10 : 20 42 59 20 48 4f 52 53  9b
ca18 : 54 20 52 45 49 43 48 45  14
ca20 : 52 54 0d 20 20 20 20 20  a8
ca28 : 20 20 20 20 20 20 20 20  28
ca30 : 20 35 39 30 30 20 53 49  23
ca38 : 45 47 45 4e 0d 00 24 23  e4
ca40 : 42 2d 50 20 31 33 20 30  bf
ca48 : 55 31 3a 31 33 20 30 20  20
ca50 : 00 00 20 00 00 42 2d 41  a2
ca58 : 20 30 20 00 00 20 00 00  9a
ca60 : 93 00 94 00 95 00 49 53  3e
ca68 : 4b 00 49 52 00 45 52 52  68
ca70 : b0 00 4d 91 00 df c2 ec  8a
ca78 : c2 dc c2 1b c3 42 c3 12  3e
ca80 : c3 ff bf 93 9a 12 44 4c  89
ca88 : 4f 41 44 20 20 0d 12 44  c8
ca90 : 53 41 56 45 20 20 0d 12  1d
ca98 : 44 56 45 52 49 46 59 0d  e9
caa0 : 12 44 49 52 20 20 20 20  35
caa8 : 0d 12 44 49 53 4b 20 20  49
cab0 : 20 0d 12 44 45 52 52 4f  33
cab8 : 52 20 0d 12 44 4d 4f 4e  29
cac0 : 20 20 20 0d 00 20 20 45  a6
cac8 : 4e 44 20 4f 46 20 46 49  3b
cad0 : 4c 45 20 21 00 93 20 54  b1
cad8 : 52 41 43 4b 20 3a 20 00  59
cae0 : 20 20 54 52 41 43 4b 20  0b
cae8 : 3a 20 00 20 53 45 4b 54  6c
caf0 : 4f 52 20 3a 20 00 20 04  42
caf8 : cb a2 00 8e 71 c9 20 b3  33
cb00 : c0 4c 1e cc 20 c1 c6 a2  78
cb08 : 29 a0 cc 20 3a c9 a2 90  56
cb10 : bd 00 cf 20 d2 ff e8 e0  58
cb18 : a0 d0 f5 20 59 c8 a2 60  c9
cb20 : a0 cc 20 3a c9 ad a2 cf  aa
cb28 : 20 d2 ff ad a3 cf 20 d2  46
cb30 : ff 20 5f c8 20 5f c8 20  91
cb38 : 5f c8 a0 14 98 20 d5 c7  17
cb40 : a9 2d 20 d2 ff 98 f0 03  71
cb48 : 20 5f c8 88 98 10 ee a2  66
cb50 : 02 a0 03 20 0c e5 a0 03  e0
cb58 : a9 30 a2 0f d0 0a c0 01  06
cb60 : f0 04 a2 10 d0 02 a2 04  ad
cb68 : 20 d2 ff ca d0 fa 18 69  63
cb70 : 01 88 d0 ea a2 03 a0 03  11
cb78 : 20 0c e5 a0 23 a9 01 85  ba
cb80 : fb 20 17 c8 20 d2 ff e6  d1
cb88 : fb a5 fb c9 10 d0 04 a9  79
cb90 : 00 85 fb 88 d0 eb a9 01  78
cb98 : 85 02 a2 04 a5 02 69 02  5c
cba0 : a8 20 0c e5 a2 14 86 8b  14
cba8 : a9 cf 85 8d a9 01 a6 02  8d
cbb0 : 69 04 ca d0 fb 85 8c a5  51
cbb8 : 8b a6 02 e0 12 90 20 e0  1b
cbc0 : 19 90 14 e0 1f 90 08 c9  6d
cbc8 : 11 90 14 a2 04 d0 2a c9  7e
cbd0 : 12 90 0c a2 04 d0 22 c9  65
cbd8 : 13 90 04 a2 04 d0 1a a5  03
cbe0 : 8b 4a 4a 4a a8 b1 8c a8  08
cbe8 : a5 8b 29 07 aa 98 3d 66  af
cbf0 : cc d0 04 a2 00 f0 02 a2  4f
cbf8 : 02 a0 02 bd 6e cc 20 d2  f6
cc00 : ff e8 88 d0 f6 a5 8b f0  5c
cc08 : 05 a9 11 20 d2 ff c6 8b  8a
cc10 : 10 a5 e6 02 a5 02 c9 24  c7
cc18 : f0 03 4c 9a cb 60 a9 00  57
cc20 : 85 c6 a5 c6 f0 fc c6 c6  ea
cc28 : 60 93 20 20 2a 3d 42 45  7e
cc30 : 4c 45 47 54 20 20 2e 3d  b1
cc38 : 46 52 45 49 20 20 a6 3d  3a
cc40 : 4c 45 45 52 20 20 20 20  8e
cc48 : 20 20 20 20 20 20 20 20  48
cc50 : 20 20 20 20 44 49 53 4b  ff
cc58 : 4e 41 4d 45 20 3a 20 00  97
cc60 : 49 44 20 3a 20 00 01 02  25
cc68 : 04 08 10 20 40 80 2a 9d  64
cc70 : 2e 9d a6 9d 60 00 00 00  d0
```

Listing »Disketten-Monitor«.
Beachten Sie bitte die Eingabehinweise auf Seite 53.

Neues von Hypra-Load

Es sind noch ein paar Bytes frei im Speicher für kleine Ergänzungen zum Hypra-Load-Perfekt. Hier eine Old-Routine und der Disk-Status-Befehl.

Hypra Perfekt ist gut, aber es geht nocht besser! Bei mir verschwinden durch RESET des öfteren Basic-Programme, oder die Floppy-LED blinkt mich fragend an. Das muß nicht sein, sagte ich mir, und untersuchte via Monitor das Betriebssystem des C 64. Nach mehreren Stunden fand ich die Routine »Laden von Band«, die ja nicht mehr benötigt wird. Auch von Hypra-Load wurde diese Routine nicht verändert beziehungsweise benötigt. Dadurch ergibt sich ein freier Speicherbereich, in den zwei kleine Programme passen: Das erste Programm (Listing 1: Status) liest den Fehlerkanal der Floppy und zeigt ihn auf dem Bildschirm an (das Programm stammt aus dem Floppy-Kurs). Das zweite Programm (Listing 1: Old) rettet ein gelöschtes Basic-Programm und setzt alle Zeiger wieder richtig. Die Old-Routine wird mit SYS62777 und die Status-Routine mit SYS62843 aufgerufen. (Ich habe mir diese Aufrufe gleich auf zwei Funktionstasten gelegt.)

Um die beiden Programme in Hypra-Perfekt einzubinden, müssen die DATA-Zeilen des Listings mit genau den gleichen Zeilennummern eingetippt werden.

Die Statusroutine belegt den Speicherplatz von 62843 bis 62894 (Dez.) und wird mit SYS 62843 aufgerufen. (Auch von einem Basic-Programm aus möglich.)

Die Old-Routine belegt den Speicherbereich von 62777 bis 62842 (Dez.) und wird mit SYS62777 aufgerufen. Nach Ablauf erfolgt ein Sprung $A474, (READY-Meldung)

(Bernhard Korell/gk)

```
8300 REM **** OLD-ROUTINE *****            <150>
8301 DATA 62777,65,7309                    <169>
8302 DATA 165,43,164,44,133,34,132,35,160,
     3,200,177,34,208,251,200,152,24,101,3
     4                                     <037>
8303 DATA 160,0,145,43,165,35,105,0,200,14
     5,43,136,162,3,230,34,208,2,230,35,17
     7                                     <033>
8304 DATA 34,208,244,202,208,243,165,34,10
     5,2,133,45,165,35,105,0,133,46        <201>
8305 DATA 32,96,166,76,116,164             <210>
8306 REM *** STATUS-ROUTINE ***            <190>
8307 DATA 62843,51,6757                    <172>
8308 DATA 169,0,133,144,169,1,162,8,160,11
     1,32,186,255,169,0,32,189,255,32,192,
     255                                   <199>
8309 DATA 169,8,32,180,255,169,111,32,150,
     255,32,165,255,32,210,255,36,144,80,2
     46                                    <209>
8310 DATA 169,8,32,171,255,169,1,32,195,25
     5                                     <136>
8350 REM *** BLOCK 17: $E49B-$E4A2         <023>
8351 DATA 58523,8,554                      <006>
8352 DATA 72,89,80,82,65,45,76,45          <117>
8355 REM *** BLOCK 17 UEBERSCHREIBT ***    <012>
8356 REM *** 64K RAM-SYSTEM IN      ***    <141>
8357 REM *** HYPRA-L-SYSTEM          ***   <086>
9000 DATA 0                                <253>
9010 REM ******  ENDE  ******              <250>
```

Listing 1. OLD und STATUS-Routine

Bildschirmmasken leicht erstellt

Das Tolle an diesem Programm ist, daß nicht wie üblich 25 PRINT-Befehle erzeugt werden. Die Bildschirmmaske wird schnell und platzsparend in Basic-Zeilen abgelegt, die FOR ... NEXT-Schleifen, POKE-, SPC- und CHR$-Befehle enthalten.

Die Idee zu diesem Programm ist nicht neu. In der 64'er-Ausgabe 9/84 wurde bereits ein Programm (allerdings für den VC 20) vorgestellt, das eine Maske direkt vom Bildschirm liest und automatisch die entsprechenden PRINT-Befehle erzeugt. Aber eben darin bestand ein Nachteil. In dem erzeugten Programm tritt nur der Befehl PRINT auf, nicht aber Befehle wie FOR, TO, NEXT, CHR$, SPC und POKE, deren Anwendung besonders dann notwendig wird, wenn der Speicherplatz nur begrenzt ist.

Verschiedene Schriftfarben innerhalb einer Maske werden im vorliegenden Programm ebenso berücksichtigt wie die Rahmen- und Hintergrundfarbe, invers dargestellte Zeichen und auch die Wahl des Zeichensatzes (Groß- oder Kleinschrift). Im Programm ist außerdem ein Editor integriert, der gegenüber dem Basic-Editor einige Verbesserung aufzuweisen hat, wodurch eine schnellere und bequemere Eingabe der Maske möglich ist.

Das Programmieren von Bildschirmmasken ist sehr zeitaufwendig, aber leider in keinem umfangreicheren Programm wegzudenken. Kompliziertere Grafiken skizziert man auf dem Papier, bevor sie in ein Programm übertragen werden. Schwierig wird es, wenn man mehrfarbige Masken programmieren will oder aber dann, wenn einige Zeichen einfach, andere dagegen invers dargestellt werden sollen.

Das Programm besteht aus einem komfortablen Editor und einem Generator, der eine eingegebene Maske innerhalb einer Sekunde in ein Basic-Programm umwandelt. Wenn Sie das schon eingetippte Programm auf einem Datenträger abgespeichert haben, kann begonnen werden.

Hinweise zum Maskengenerator

Das Programm wird mit LOAD "Programmname", x,1 (x = Gerätenummer) in den Speicher geladen. Da es sich hier um ein Maschinenprogramm handelt, das die oberen 4 KByte RAM belegt, wird ein im Basic-Speicher stehendes Programm nicht gelöscht.

Mit SYS 49152,ln,st wird der Maskengenerator gestartet. Das noch zu erzeugende Programm steht nachher ab Zeile ln im Speicher. Beachten Sie, daß, wenn bereits ein Programm im Speicher steht, der Parameter ln größer als die letzte Zeilennummer sein muß. Die Durchnumerierung des zu erzeugenden Programms erfolgt mit der Schrittweite st (0 < st < 256).

Geben Sie nun eine Maske ein. Nehmen Sie dazu die Tabelle 1 der Steuerbefehle zu Hilfe. Neben diesen Steuerbefehlen kann man auf gewohnte Weise jede der 16 zur Verfügung stehenden Schriftfarben anwählen. Alle Zeichen und Buchstaben können uneingeschränkt verwendet werden; das gilt auch für das Anführungszeichen.

Ist die Eingabe beendet, dann geben Sie RUN/STOP und Y ein. Innerhalb einer Sekunde wird nun das Maskenprogramm erzeugt. Mit RUN ln können Sie es wie gewohnt starten.

(Georg Wichert/ah)

Bildschirmmaskengenerator — Beschreibung der Steuerbefehle

ASCII-Code	Eingabe	Bedeutung/Wirkung
3	CTRL-C STOP	Inhalt der Cursorzeile wird in die Mitte verschoben. Eingabe beenden. Sicherheitsabfrage »finished?« mit Y (= Yes) oder N (= No) beantworten.
4	CTRL-D	Cursorzeile und die Zeilen darunter werden um eine Zeile nach oben verschoben. Letzte Zeile wird gelöscht.
9	CTRL-I	Cursorzeile und die Zeilen darunter werden um eine Zeile nach unten verschoben. Cursorzeile wird gelöscht.
13	RETURN	Cursor springt zum Anfang der nächsten Zeile.
17	CRSR down	Cursor geht um eine Zeile nach unten. In der letzten Zeile Sprung in die erste.
19	HOME	Sprung in die linke obere Ecke
20	DEL	Das links vom Cursor stehende Zeichen wird gelöscht und der Rest der Zeile um eine Position nach links verschoben.
24	CTRL-X	Cursorzeile wird gelöscht
25	CTRL-Y	Cursorspalte wird gelöscht
29	CRSR right	Cursor geht um eine Position nach rechts.
133	F1	Rahmenfarbe verändern.
134	F3	Hintergrundfarbe verändern.
135	F5	Dauerfunktion aller Tasten ein- oder ausschalten.
136	F7	RVS-Modus an- oder ausschalten.
145	CRSR up	Cursor geht um eine Zeile nach oben. In der ersten Zeile Sprung in die letzte.
147	CLR	Bildschirm löschen. Sicherheitsabfrage »clear screen?« mit Y (= Yes) oder N (= No) beantworten.
148	INST	Cursorposition und der Rest der Zeile werden innerhalb einer Zeile um eine Position nach rechts geschoben. Ein Leerzeichen wird eingefügt.
157	CRSR left	Cursor geht um eine Position nach links.
	Commodore- und CTRL	Umschaltung Groß-/Kleinschrift

Tabelle 1. Editor-Steuerbefehle zum Maskengenerator

```
programm : maskengen.      c000 c5bd
-------------------------------------
c000 : 20 fd ae 20 eb b7 e0 00  ce
c008 : d0 03 4c 08 af 86 02 a5  f0
c010 : 15 c9 fa b0 f5 20 13 a6  d8
c018 : a0 01 b1 5f d0 ec 20 5e  43
c020 : e5 a9 00 85 c7 20 e4 ff  9c
c028 : 85 cc f0 f9 48 a5 cf f0  61
c030 : 0c a5 ce ae 87 02 a0 00  a3
c038 : 84 cf 20 13 ea 68 c9 20  68
c040 : 90 2f c9 61 b0 0d 29 3f  9d
c048 : 05 c7 ae 86 02 20 13 ea  f1
c050 : 4c 96 c1 c9 80 b0 04 29  81
c058 : df d0 ed c9 a0 90 12 c9  be
c060 : c0 b0 04 e9 3f d0 e1 29  0b
c068 : 7f c9 7f d0 db a9 5e d0  ec
c070 : d7 20 cb e8 10 af a0 33  cf
c078 : d9 8d c0 f0 07 88 88 88  4e
c080 : 10 f6 30 a1 b9 8f c0 48  f7
c088 : b9 8e c0 48 60 03 c2 c0  6c
c090 : 04 05 c1 09 25 c1 0d 4a  d2
c098 : c1 11 5d c1 13 67 c1 14  0d
c0a0 : 6c c1 18 78 c1 19 7f c1  68
c0a8 : 1d 95 c1 85 c0 c1 86 c3  6d
c0b0 : c1 87 b0 c1 88 d1 c1 91  da
c0b8 : da c1 93 e6 c1 94 19 c2  df
c0c0 : 9d 21 c2 a5 91 c9 7f d0  5a
c0c8 : 03 4c b3 c2 20 30 c2 84  ce
c0d0 : 06 a9 27 38 e5 06 85 06  2c
c0d8 : 30 29 a0 00 b1 d1 c9 20  d6
c0e0 : d0 05 c8 c0 28 d0 f5 98  8f
c0e8 : 84 04 18 65 06 4a aa a0  c0
c0f0 : 00 e4 04 f0 0e b0 06 20  40
c0f8 : 43 c2 e8 10 f2 20 61 c2  14
c100 : ca 10 ec 4c 25 c0 a6 d6  38
c108 : f0 f9 ca 86 d6 20 f0 e9  7e
c110 : bd f1 ec 85 ac b5 da 20  d6
c118 : c8 e9 e8 e0 18 90 ee 20  2d
c120 : ff e9 a6 d6 10 32 a5 d1  65
c128 : 48 a5 d2 48 a2 18 e4 d6  2d
c130 : f0 ed 20 f0 e9 bd ef ec  63
c138 : 85 ac b5 d8 20 c8 e9 ca  21
c140 : 10 ec 68 85 d2 68 85 d1  bb
c148 : 4c 25 c0 a6 d6 e0 18 f0  e2
c150 : 0a a9 00 85 d3 e8 86 d6  2c
c158 : 20 f0 e9 4c 25 c0 a6 d6  95
c160 : e0 18 d0 f1 a2 00 f0 ee  8a
c168 : 20 66 e5 30 ee a4 d3 f0  80
c170 : ea 88 84 d3 20 43 c2 10  81
c178 : e2 a6 d6 20 ff e9 30 db  2f
c180 : a5 d1 48 a5 d2 48 a2 18  ff
c188 : ac d3 00 20 f0 e9 20 57  af
c190 : c2 ca 10 f7 30 ac a9 27  18
c198 : c5 d3 d0 10 a6 d6 e0 18  52
c1a0 : f0 c6 e8 86 d6 20 f0 e9  04
c1a8 : a9 ff 85 d3 e6 d3 4c 25  b5
c1b0 : c0 ad 8a 02 30 03 09 80  6a
c1b8 : 2c 29 7f 8d 8a 02 4c 25  3f
c1c0 : c0 a0 00 2c a0 01 be 20  a3
c1c8 : d0 e8 8a 29 0f 99 20 d0  b4
c1d0 : 10 dc 06 c7 b0 d8 38 66  48
c1d8 : c7 90 d3 a6 d6 f0 04 ca  4c
c1e0 : 4c 56 c1 a2 18 d0 f9 a0  4d
c1e8 : 00 be 0b c2 f0 05 20 7b  11
c1f0 : c2 d0 f6 8c 06 00 20 e4  14
c1f8 : ff f0 fb c9 59 f0 09 c9  7c
c200 : 4e d0 f3 20 93 c2 30 a6  15
c208 : 4c 1e c0 83 8c 85 81 92  24
c210 : a0 93 83 92 85 85 8e a0  ad
c218 : bf 00 a4 d3 20 61 c2 4c  2c
c220 : 25 c0 a4 d3 d0 06 a0 27  57
c228 : 84 d3 d0 af c6 d3 10 ef  eb
c230 : a0 27 b1 d1 c9 20 f0 04  74
c238 : c9 60 d0 05 88 10 f3 18  0f
c240 : 60 38 60 20 24 ea c8 b1  f9
c248 : d1 88 91 d1 c8 b1 f3 88  f7
c250 : 91 f3 c8 c0 27 90 ef a9  2f
c258 : 20 91 d1 ad 86 02 91 f3  11
c260 : 60 20 24 ea 84 05 a0 27  78
c268 : 88 b1 d1 c8 91 d1 88 b1  83
c270 : f3 c8 91 f3 88 c4 05 d0  0e
c278 : ef f0 dc b9 00 04 99 34  3d
c280 : 03 b9 00 d8 99 84 03 8a  5a
c288 : 99 00 04 ad 86 02 99 00  b7
c290 : d8 c8 60 ac 06 00 88 30  5d
c298 : f9 b9 84 03 99 00 d8 b9  60
c2a0 : 34 03 99 00 04 4c 96 c2  3f
c2a8 : 86 89 8e 89 93 88 85 84  64
c2b0 : a0 bf 00 a0 00 be a8 c2  62
c2b8 : f0 05 20 7b c2 d0 f6 8c  4a
c2c0 : 06 00 20 e4 ff f0 fb c9  76
c2c8 : 59 f0 0a c9 4e d0 f3 20  d1
c2d0 : 93 c2 4c 25 c0 20 93 c2  5d
c2d8 : 78 a2 00 a0 04 20 24 c5  13
c2e0 : a0 0c ad 20 d0 29 0f aa  de
c2e8 : 20 65 c5 a0 17 ad 21 d0  45
c2f0 : 29 0f aa 20 65 c5 ad 18  bb
c2f8 : d0 4a 4a b0 06 a0 20 a9  cf
c300 : 32 91 5f a0 29 20 86 c5  20
c308 : a2 ff 86 d6 8e 86 02 e8  1d
c310 : 86 07 86 09 06 c7 a9 00  22
c318 : 85 08 85 d3 e6 d6 a6 d6  ea
c320 : e0 19 90 03 4c bc c3 20  0b
c328 : f0 e9 20 30 c2 b0 04 e6  aa
c330 : 08 d0 e9 84 03 a6 08 f0  13
c338 : 0f 20 e3 c3 a4 05 c0 04  46
c340 : f0 03 20 86 c5 20 f5 c4  49
c348 : 20 24 ea a4 d3 [illegible] [illegible] 29  b6
c350 : 0f cd 86 02 f0 25 aa b1  6e
c358 : d1 c9 20 f0 20 c9 60 f0  e8
c360 : 1c 8e 86 02 20 e3 c3 20  16
c368 : 14 c5 98 ac 86 02 be da  44
c370 : e8 a8 8a 91 5f c8 84 05  d9
c378 : ac d3 00 b1 d1 a6 09 f0  9c
c380 : 09 c5 ce f0 27 48 20 e3  3a
c388 : c3 68 85 ce 24 ce 30 0b  4a
c390 : 06 c7 90 18 20 14 c5 a9  ae
c398 : 92 d0 0c 24 c7 30 0d 38  bd
c3a0 : 66 c7 20 14 c5 a9 12 91  8a
c3a8 : 5f c8 84 05 e6 09 e6 d3  27
c3b0 : a5 03 c5 d3 b0 95 20 e3  c3
c3b8 : c3 4c 16 c3 20 e3 c3 a4  19
c3c0 : 05 c0 04 f0 03 20 86 c5  1b
c3c8 : a9 00 a0 01 91 5f 88 10  10
c3d0 : fb a5 5f 18 69 02 8d 2d  b0
c3d8 : 00 a5 60 69 00 8d 2e 00  15
c3e0 : 4c 6c fe a6 09 d0 01 60  d3
c3e8 : a4 05 a5 ce 29 7f c9 20  48
c3f0 : 90 09 c9 40 90 0a c9 60  c0
c3f8 : 90 04 18 69 40 2c 69 20  09
c400 : 85 ce c9 22 d0 30 e0 05  bf
c408 : b0 1c 20 ae c4 86 06 a2  82
c410 : 2f 20 24 c5 c6 06 d0 f7  e1
c418 : 84 05 a6 03 e4 d3 b0 49  6b
c420 : e0 27 f0 3c d0 3e 20 d0  79
c428 : c4 a2 2f 20 24 c5 20 c7  8e
c430 : c4 20 ee c4 d0 66 c9 20  00
c438 : f0 34 c9 a0 f0 30 e0 14  05
c440 : b0 46 20 14 c5 a5 ce 91  86
c448 : 5f c8 ca d0 fa 84 05 a6  0d
c450 : 03 e4 d3 b0 14 e0 27 d0  57
c458 : 0b c0 14 90 0c 20 b7 c4  05
c460 : 20 c7 c4 2c 06 c7 20 86  47
c468 : c5 a9 00 85 09 60 24 c7  66
c470 : 30 cc e0 06 90 cc 20 ae  4c
c478 : c4 20 c1 c4 20 65 c5 a9  ed
c480 : 29 91 5f c8 84 05 d0 e1  da
c488 : 20 d0 c4 20 be c4 a5 ce  8c
c490 : 91 5f c8 20 be c4 20 c7  29
c498 : c4 20 ee c4 a6 03 e4 d3  7e
c4a0 : b0 c4 e0 27 f0 c0 20 c4  ee
c4a8 : c4 20 ca c4 d0 b6 c0 04  95
c4b0 : f0 18 06 07 b0 08 60 06  e8
c4b8 : 07 b0 03 38 66 07 a9 22  69
c4c0 : 2c a9 a6 2c a9 3a 2c a9  60
c4c8 : 3b 2c a9 99 91 5f c8 60  af
c4d0 : c0 04 f0 09 20 b2 c4 20  db
c4d8 : c7 c4 20 c4 c4 86 06 a2  80
c4e0 : 26 20 24 c5 a6 06 20 65  be
c4e8 : c5 20 c4 c4 d0 dc 20 c4  85
c4f0 : c4 a9 82 d0 d7 06 c7 a6  5e
c4f8 : 08 e0 06 b0 0f 20 ca c4  af
c500 : ca f0 05 20 c4 c4 d0 f5  29
c508 : 20 86 c5 60 20 d0 c4 20  c5
c510 : ee c4 d0 f4 a4 05 c0 04  b0
c518 : d0 05 20 ca c4 d0 9c 24  5a
c520 : 07 10 98 60 bd 30 c5 f0  b8
c528 : 06 91 5f c8 e8 d0 f5 60  95
c530 : 97 20 35 33 32 38 30 2c  89
c538 : 20 20 3a 97 20 35 33 32  c7
c540 : 38 31 2c 20 20 3a 99 20  9a
c548 : c7 28 31 34 20 29 c7 28  b1
c550 : 31 34 37 29 3b 00 81 20  88
c558 : 5a b2 31 20 a4 20 00 c7  37
c560 : 28 33 34 29 00 84 05 a9  e0
c568 : 00 85 62 86 63 a2 90 38  92
c570 : 20 49 bc 20 df bd a2 00  de
c578 : a4 05 bd 00 01 f0 06 91  e1
c580 : 5f e8 c8 d0 f5 60 a9 00  a8
c588 : 91 5f 84 06 a0 02 a5 14  84
c590 : 91 5f 18 65 02 85 14 a5  6c
c598 : 15 90 02 e6 15 c8 91 5f  ef
c5a0 : a0 00 a5 5f 38 65 06 91  80
c5a8 : 5f 48 a5 60 69 00 c8 91  7e
c5b0 : 5f 85 60 68 85 5f a0 04  d5
c5b8 : 84 05 06 07 60 ff ff ff  27
```

Listing Maskengenerator. Bei der Eingabe bitte den MSE verwenden und Seite 53 beachten

Maschinensprache »getürkt«

Es gibt eine einfache Methode, um beim VC 20 mit mindestens 8 KByte Erweiterung aus einem ganz normalen Basic-Programm scheinbar ein Maschinensprache-Programm zu machen.

Zunächst gibt man die folgende Zeile ein:

10 SYS 4621

Danach folgt im Direktmodus:

POKE 43, 1 : POKE 44, 19 : POKE 4864, 0 : NEW

Keine Angst, die Programmzeile 10 ist nicht verloren. Jetzt muß das folgende kleine Programm eingegeben werden.

```
10 FOR A = 4621 TO 4643
20 READ D : S = S + D : POKE A, D
30 NEXT
40 IF S<>2039 THEN PRINT "FEHLER IN DATAS"
50 DATA 169, 36, 133, 43, 169, 18, 133, 44
60 DATA 169, 0, 141, 34, 3, 141, 35, 3, 32
70 DATA 89, 198, 76, 174, 199, 0
```

Dieses Programm wird jetzt mit »RUN« gestartet. Anschließend wird im Direktmodus der Basic-Start verschoben:

POKE 43, 36 : POKE 44, 18 : POKE 4643, 0 : NEW

Damit sind wir schon fast am Ziel. Jetzt kann ein beliebiges Basic-Programm geladen oder geschrieben werden. Die letzte ausführbare Zeile dieses Programms muß immer lauten:

»(Zeilennummer)« POKE 43, 1 : POKE 44, 18 : END

Um das Programm abspeichern zu können, muß man jetzt noch »POKE 43, 1 : POKE 44, 18« eingeben. Gibt man nun »LIST« ein, so sieht man nur »10 SYS 4621«. Das auf solch wundersame Art zum »Maschinenprogramm« gewordene Basic-Programm kann jetzt ganz normal gespeichert, geladen und gestartet werden. Allerdings ist jetzt ein hervorragender Listschutz realisiert, zumal im Maschinenspracheteil des Programms noch die RUN/STOP-Taste gesperrt wird. Ein Versuch, das Programm damit abzubrechen, führt stets zu einem völligen »Absturz« des Computers. (A. Kaminski/ev)

Checksummer 64 mit akustischer Anzeige

Beim Abtippen eines Programms mittels Checksummer hat es mich gestört, daß man die Prüfziffer optisch kontrollieren muß und dabei den Blickkontakt zur Vorlage verliert, das heißt die nächste Zeile wieder suchen muß.

Ich habe deshalb das Programm derart erweitert, daß die richtige oder falsche Eingabe zusätzlich durch einen hohen oder einen tiefen Ton angezeigt wird. Dazu ist es erforderlich, nach dem letzten Zeichen der Zeile einen Doppelpunkt und unmittelbar danach die Prüfziffer einzutippen. Erst dann wird mit RETURN abgeschlossen und die Zeile wird, selbstverständlich ohne Doppelpunkt und Prüfziffer, abgespeichert. Ein tiefer Ton signalisiert einen Fehler, ein hoher Ton bedeutet richtige Eingabe. Die Anzeige der Prüfziffer links oben erfolgt nach wie vor. Es ist auch möglich, auf das Eintippen der Prüfziffer zu verzichten, dann erfolgt keine akustische Anzeige.

Der erste Teil des Maschinenprogramms ist im Speicher von dezimal 679 bis 767 abgelegt und ist für die Überprüfung, ob die letzte Eingabe eine Prüfziffer nach dem Doppelpunkt war oder nicht, verantwortlich. Der zweite Teil ist im Kassettenpuffer von dezimal 828 bis 906 untergebracht und bewirkt die entsprechende Tonausgabe.

Folgende Zeilen wären in das Checksummerprogramm einzufügen beziehungsweise zu ändern (Zeilen 210 und 410):

```
152   poke 42359,167:poke 42360,2
155   poke 251,252:poke 252,0
170   for i = 828 to 906:read a:poke i,a:next i
180   for i = 679 to 767:read a:poke i,a:next i
210   print"(down)anschalten(2 space): poke1,53"
220   print"(2 down)am ende der zeile doppelpunkt und die prüfzif-
      fer eingeben!":new
410   data 255,76,60,3,92,72,32,201
600   data 169,0,205,238,3,240,69,169,15,141,24,212,169,48,
      141,6,212,169,16
601   data 141,5,212,164,253,196,2,240,18,169,9,141,1,212,169,
      247,141,0,212
602   data 169,33,141,4,212,76,121,3,169,64,141,1,212,169,191,
      141,0,212,169
603   data 17,141,4,212,162,255,160,255,136,208,253,202,208,
      248,162,0,142,24
604   data 212,76,128,164
700   data 134,250,202,160,0,132,254,140,238,3,132,255,189,0,
      2,201,48,144,65
701   data 201,59,176,61,201,58,240,14,56,233,48,200,192,4,
      240,49,145,251,202
702   data 76,179,2,134,250,160,0,132,41,132,114,164,255,132,
      40,160,100,140
703   data 238,3,132,113,32,87,179,134,255,166,254,134,40,
      162,10,134,113,32
704   data 87,179,138,24,101,255,101,253,133,253,166,250,76,
      202,170
```

Mit diesen Änderungen ist der Checksummer 64 noch um einiges bedienerfreundlicher geworden. (Karl Heinz Hödl)

ROM-Modul-Schalter für C 64

Vor einiger Zeit habe ich mir Simons Basic als ROM-Modul zugelegt, um das lästige Laden von Diskette zu vermeiden. Es stellte sich jedoch bald ein gravierender Nachteil heraus: Viele Hilfsprogramme laufen nicht bei eingestecktem Modul, da der Speicherbereich $8000 - $9FFF durch Simons Basic blockiert wird.

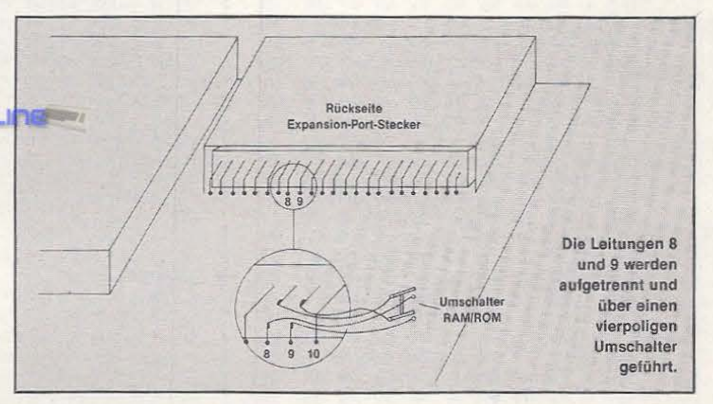
Dadurch wird es notwendig, beim Arbeiten mit derartigen Programmen jedesmal das Steckmodul zu entfernen. Dieses dauernde Heineinstecken und Herausziehen ist nicht nur lästig, sondern auf lange Sicht auch den Kontakten nicht besonders zuträglich. Besser wäre der Einbau eines Umschalters, der dem Computer das Entfernen des Moduls vorspiegelt.

Die Anwesenheit eines ROM-Moduls am Expansion-Port wird dem C 64 durch zwei Signalleitungen (EXROM und GAME) angezeigt. Liegen beide Leitungen auf »High«, also auf logisch Eins, dann wird der normale RAM-Bereich benutzt. Befinden sich diese Leitungen jedoch auf »Low«, also auf logisch Null, dann wird der externe ROM-Bereich des Moduls adressiert. Normalerweise ist es nun so, daß die beiden Leitungen durch das eingesteckte Modul selbst auf »Low« gezogen werden und somit eine softwaremäßige Änderung des Zustandes nicht mehr möglich ist.

Ich habe deshalb die beiden Leitungen (Kontakte 8 und 9, siehe Bild) hinter dem Expansion-Port, also innerhalb des C 64, aufgetrennt und zu einem zweipoligen Umschalter an der Außenseite des Computers geführt. Das Anbringen des Schalters am Oberteil des Gehäuses stellt kein Problem dar. Lediglich die Unterbrechung der Leitungen 8 und 9 direkt hinter dem Expansion-Port ist etwas problematisch, da die Anschlußleitungen zur Hauptplatine sehr dicht beieinander liegen. Aus dem gleichen Grunde ist auch das Anlöten der Verbindungsdrähte zum Umschalter eine etwas knifflige Arbeit.

Mit dieser Schaltung ist es nun möglich, bei ständig eingesetztem Modul zwischen internem RAM und externem ROM umzuschalten. Das Umschalten muß jedoch vor dem Einschalten des Computers geschehen, oder der C 64 muß nach dem Umschalten kurz aus- und wieder eingeschaltet werden.

Es versteht sich wohl von selbst, daß dieser Eingriff nur nach Ablauf der Garantiefrist für den C 64 vorgenommen werden darf. (Kurt Pfahl)



Die Leitungen 8 und 9 werden aufgetrennt und über einen vierpoligen Umschalter geführt.

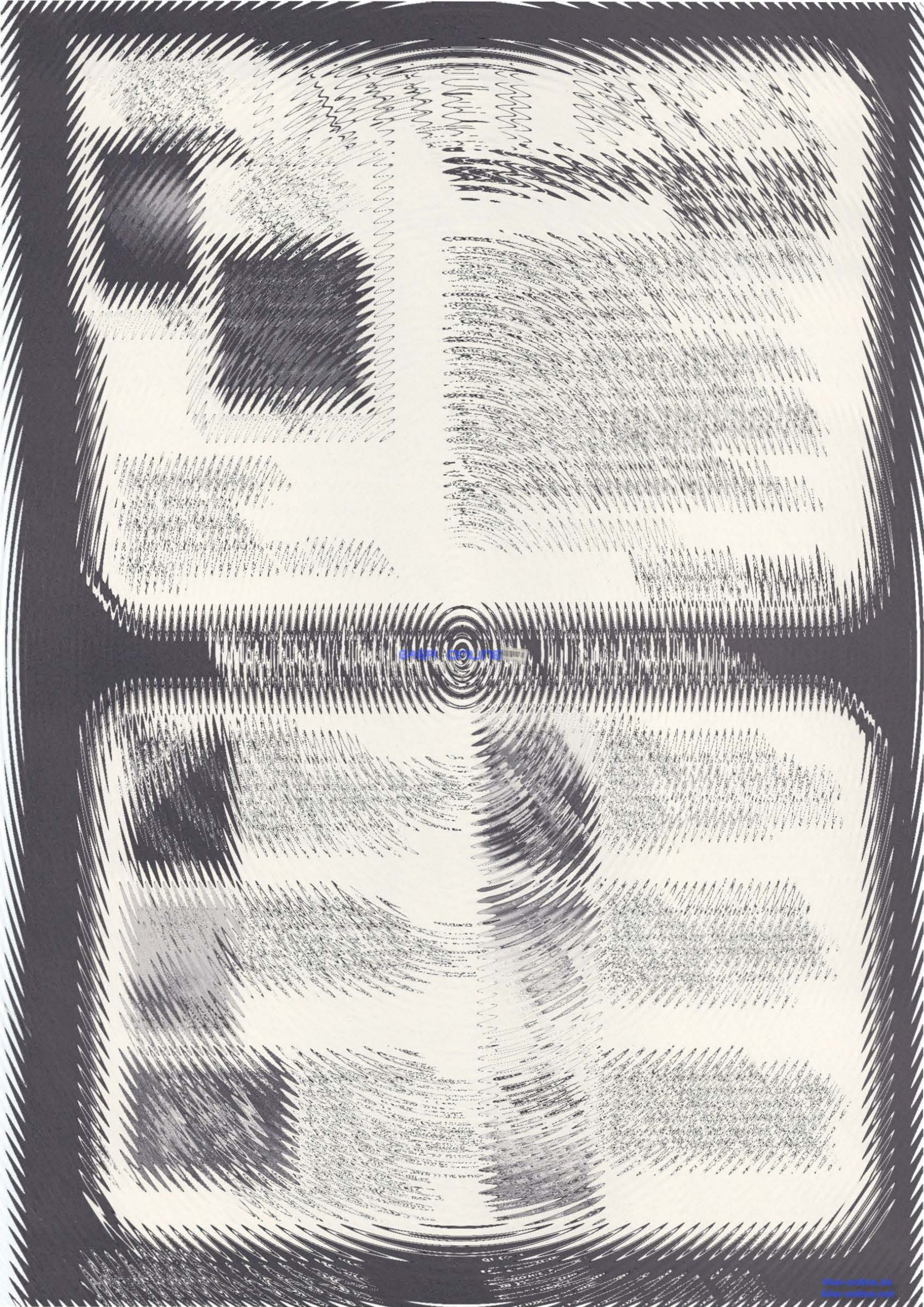
1520-Hardcopy verbessert

Über das Maschinenprogramm »Farbige 1520-Hardcopy« aus der Ausgabe 10/84, habe ich mich als neuer Besitzer eines 1520-Printer-Plotters zunächst sehr gefreut. Nach dem mühsamen Eintippen der DATA-Zeilen war aber die Enttäuschung über das Mini-Bild groß. Nachdem das 1520-Format sowieso schon recht schmal ist, sollte man es doch wenigstens ausnutzen!

Gleiches gilt für die einfarbige 1520-Hardcopy-Routine in Basic, die in der Ausgabe 7/84 veröffentlicht wurde. Nachdem ich kein Maschinensprache-Profi bin, mußte allerdings eben diese Routine herhalten. Durch Ändern von vier Zeilen liefert sie brauchbare Hardcopies im 1:2 Querformat:

```
460 X=SW-C-SQ+Z+70 : Y=-L : PRINT#1, "M", -2*Y,
2*X
510 IF B(Z+1)=0 THEN PRINT#1, "D", -2*Y, 2*X+1 :
PRINT#1, "D", -2*Y+1, 2*X+1
511 PRINT#1, "D", -2*Y+1, 2*X : GOTO 454
520 Z=Z+1 : X=X+1 : GOTO 510
```

Ein Problem ist natürlich die sehr niedrige Geschwindigkeit des Basic-Programms. Aber vielleicht kommt ja noch mal eine vierfarbige 1520-Hardcopy-Routine für Querformat in Maschinensprache. Bis dahin aber dürfte das Basic-Programm aus Ausgabe 7/84 mit den eben beschriebenen Änderungen langsame, aber gute Dienste leisten. (Elmar Walter)



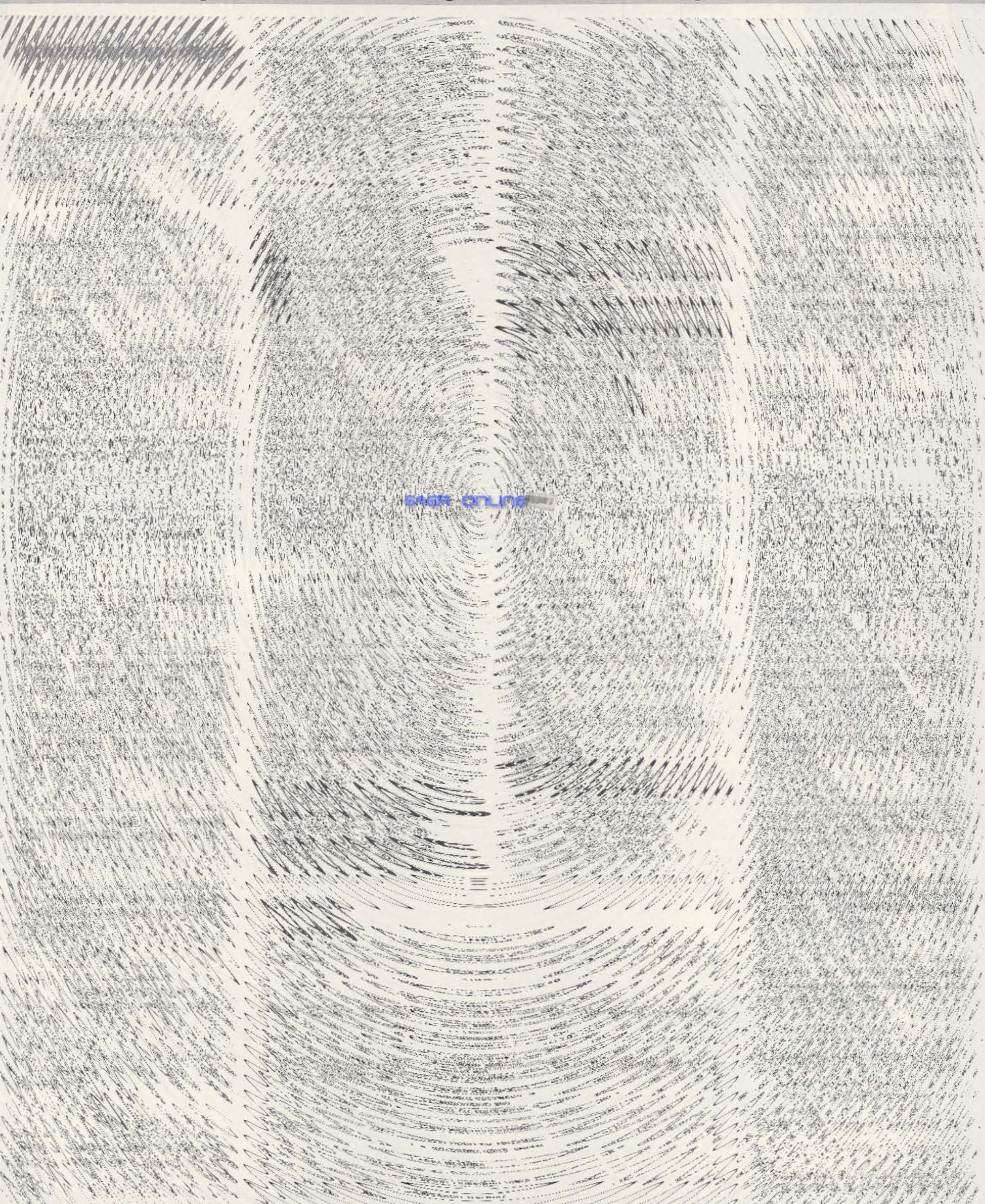
COMPUTER-MARKT

Wollen Sie einen gebrauchten Computer verkaufen oder erwerben? Suchen Sie Zubehör? Haben Sie Software anzubieten oder suchen Sie Programme oder Verbindungen? Der COMPUTER-MARKT von »64er« bietet allen Computerfans die Gelegenheit, für nur 5,— DM eine private Kleinanzeige mit bis zu 5 Zeilen Text in der Rubrik Ihrer Wahl aufzugeben. Und so kommt Ihre private Kleinanzeige in den COMPUTER-MARKT der **September-Ausgabe** (erscheint am 16. August 85): Schicken Sie Ihren Anzeigentext bis zum 22. Juli 85 (Eingangsdatum beim Verlag) an »64er«. Später eingehende Aufträge werden in der **Oktober-Ausgabe** (erscheint am 20. September 85) veröffentlicht.

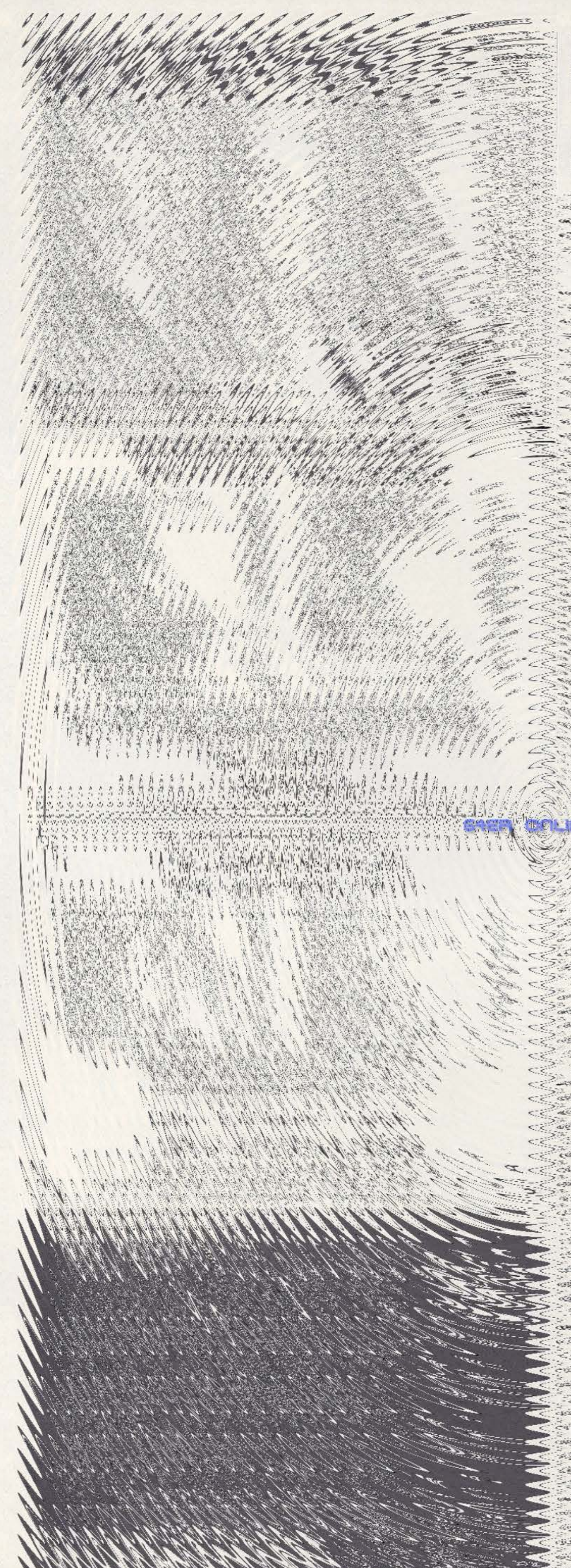
Am besten verwenden Sie dazu die vorbereitete Auftragskarte am Anfang des Heftes.
Bitte beachten Sie: Ihr Anzeigentext darf maximal 5 Zeilen mit je 32 Buchstaben betragen.
Überweisen Sie den Anzeigenpreis von DM 5,— auf das Postscheckkonto Nr. 14199-803 beim Postscheckamt mit dem Vermerk »Markt & Technik, 64er« oder schicken Sie uns DM 5,— als Scheck oder in Bargeld. Der Verlag behält sich die Veröffentlichung längerer Texte vor. Kleinanzeigen, die entsprechend gekennzeichnet sind, oder deren Text auf eine gewerbliche Tätigkeit schließen läßt, werden in der Rubrik »Gewerbliche Kleinanzeigen« zum Preis von DM 11,— je Zeile Text veröffentlicht.

Private Kleinanzeigen

Private Kleinanzeigen Private Kleinanzeigen Private Kleinanzeigen

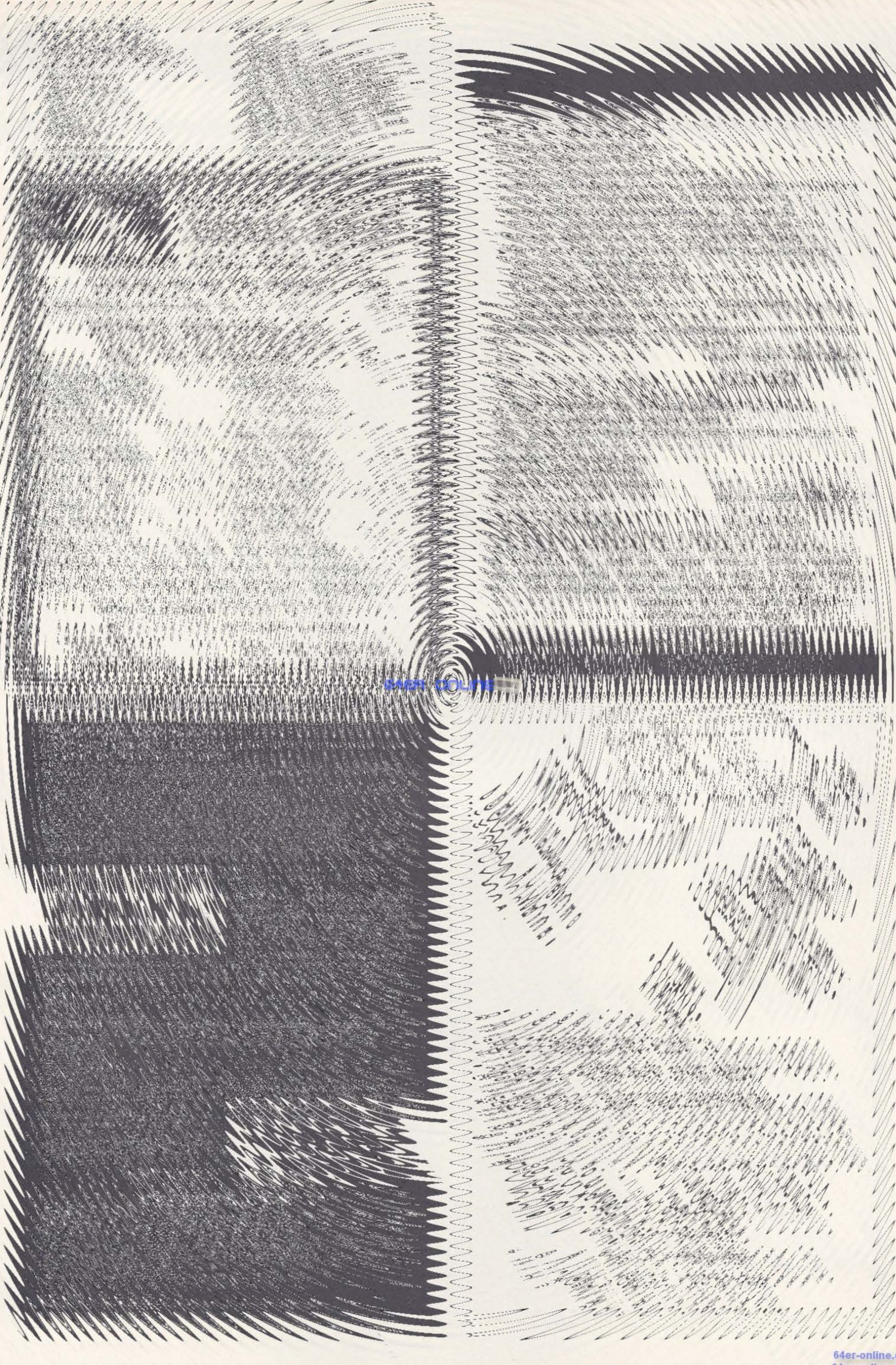




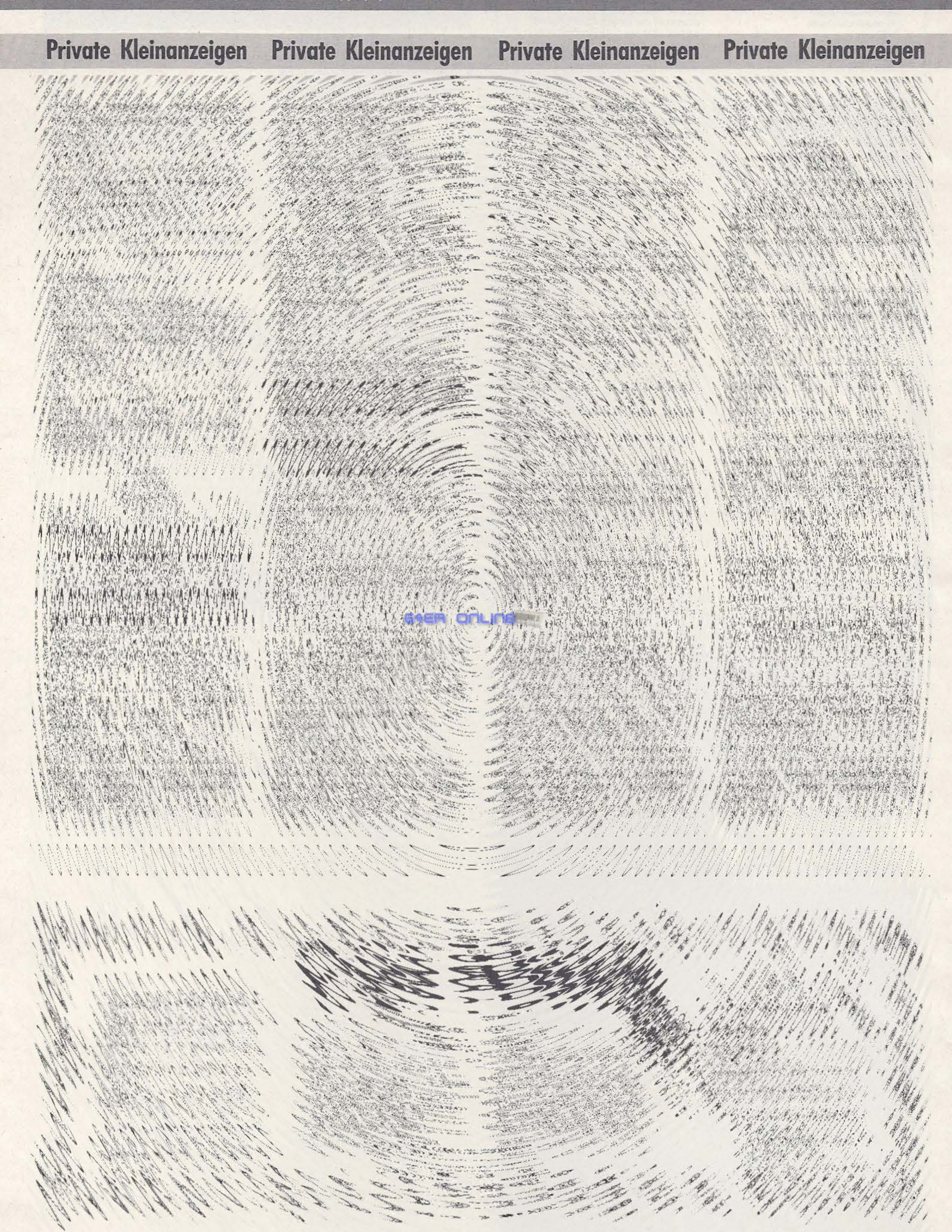


COMPUTER-MARKT

Private Kleinanzeigen Private Kleinanzeigen

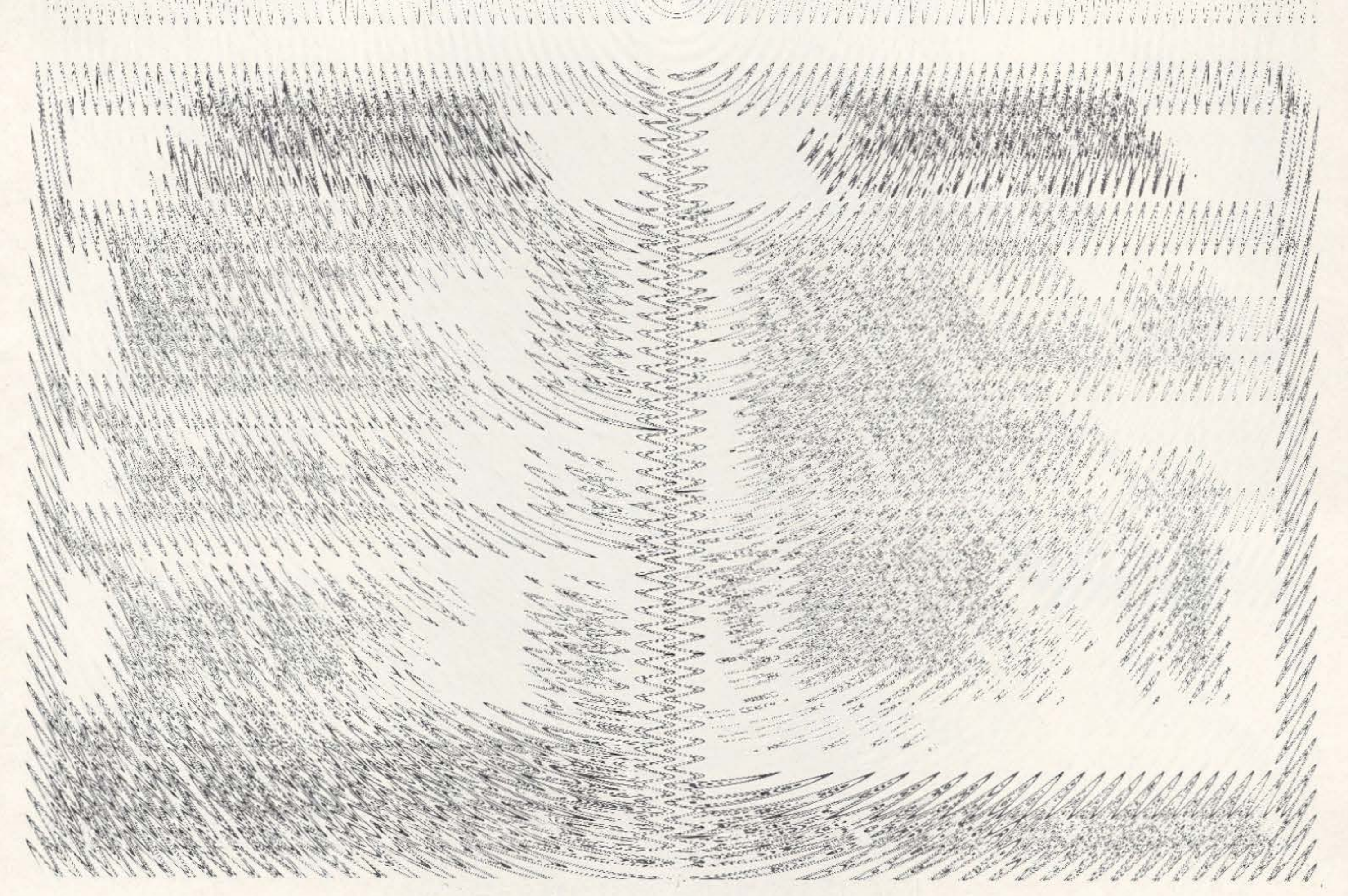


Private Kleinanzeigen Private Kleinanzeigen Private Kleinanzeigen Private Kleinanzeigen



Private Kleinanzeigen

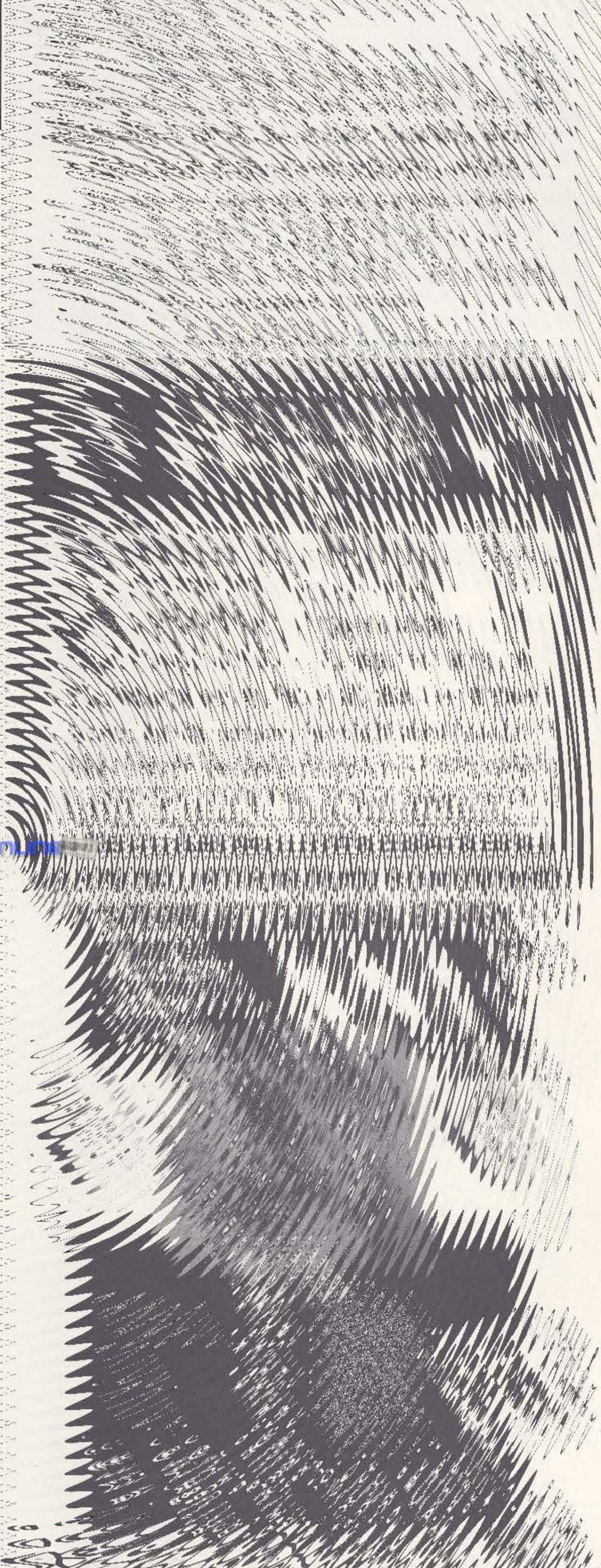
Private Kleinanzeigen Private Kleinanzeigen Private Kleinanzeigen





Private Kleinanzeigen Private Kleinanzeigen Private Kleinanzeigen Private Kleinanzeigen





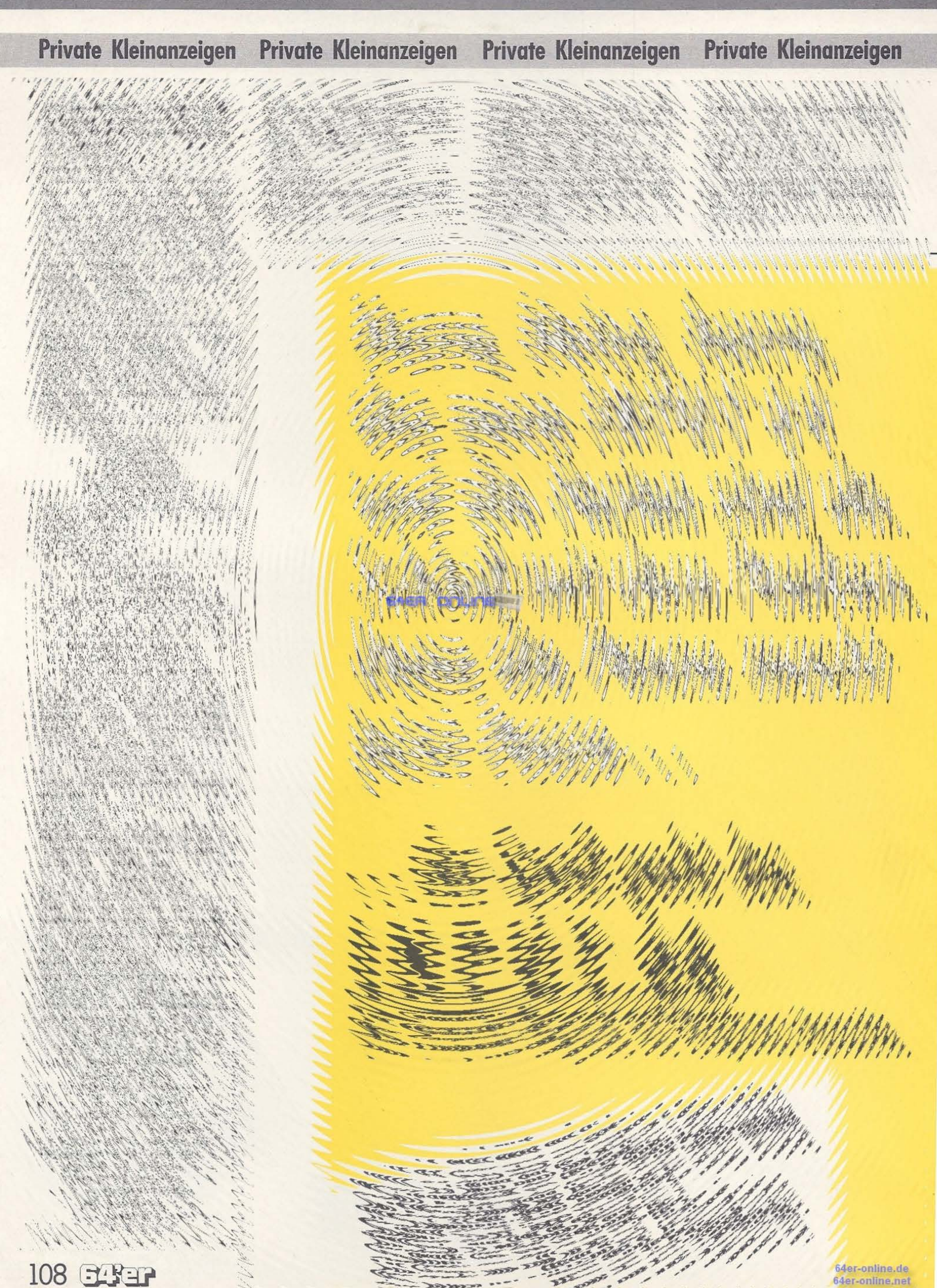






Private Kleinanzeigen

Private Kleinanzeigen Private Kleinanzeigen Private Kleinanzeigen Private Kleinanzeigen



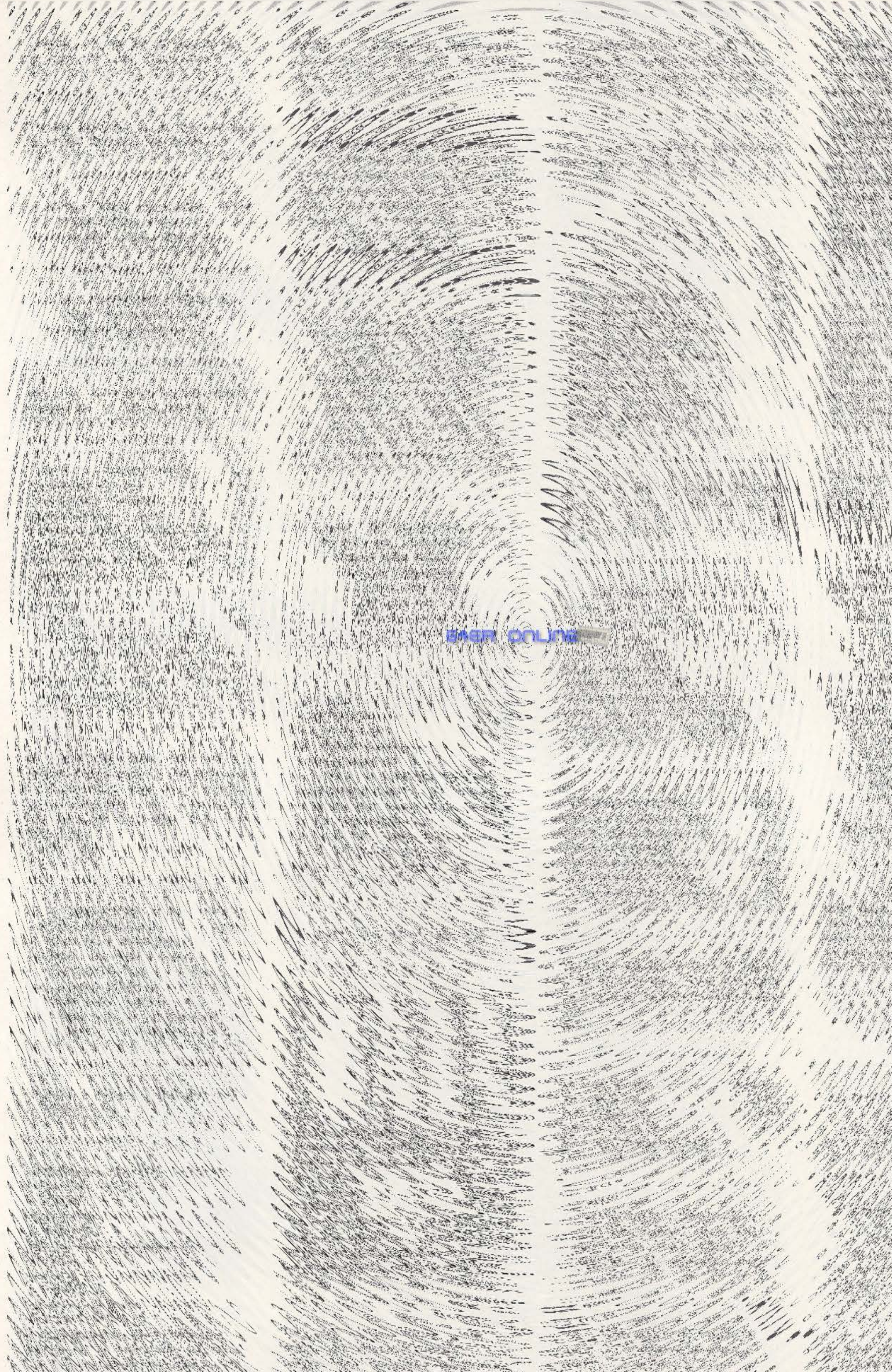


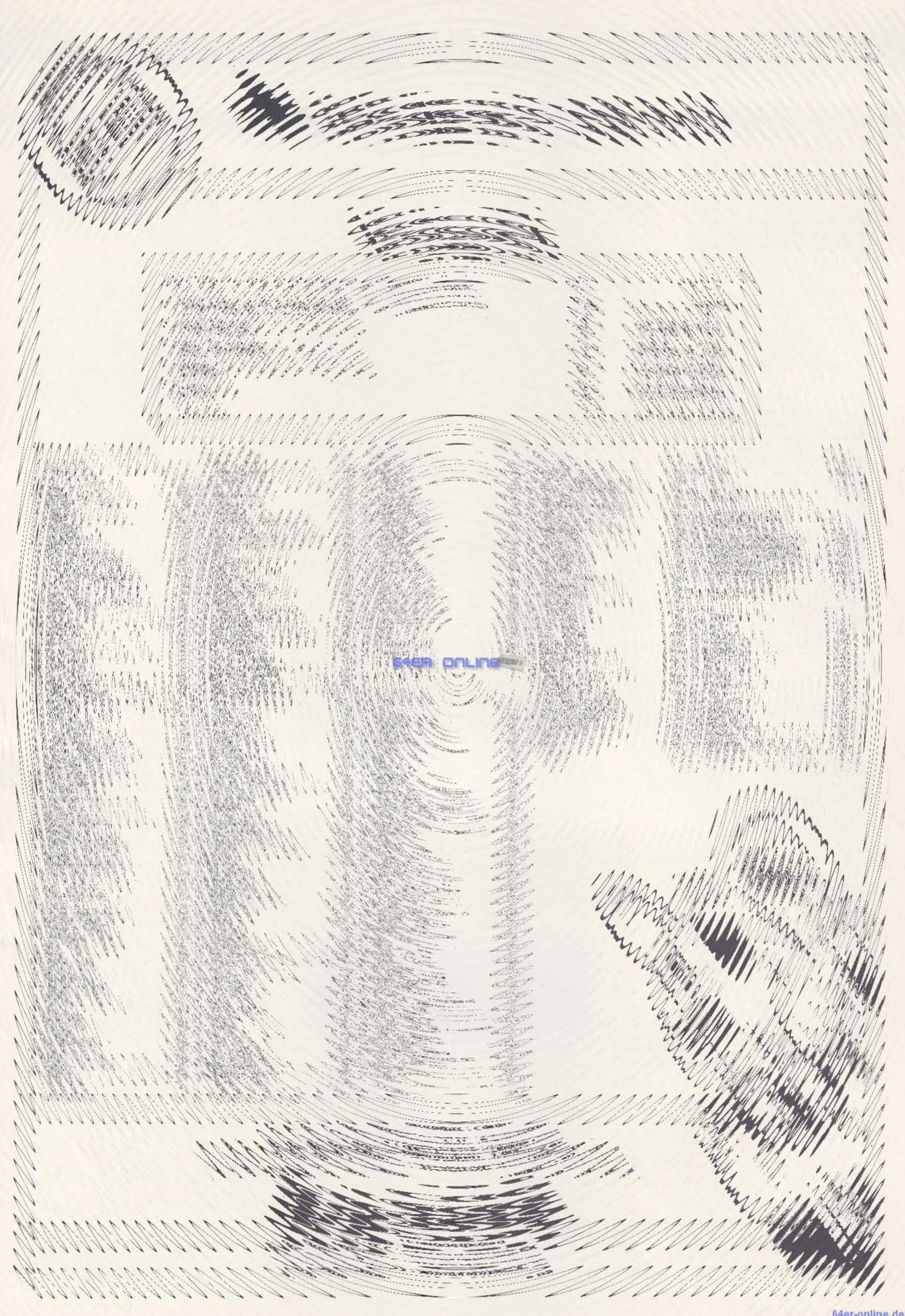
Private Kleinanzeigen Private Kleinanzeigen



Gewerbliche Kleinanzeigen

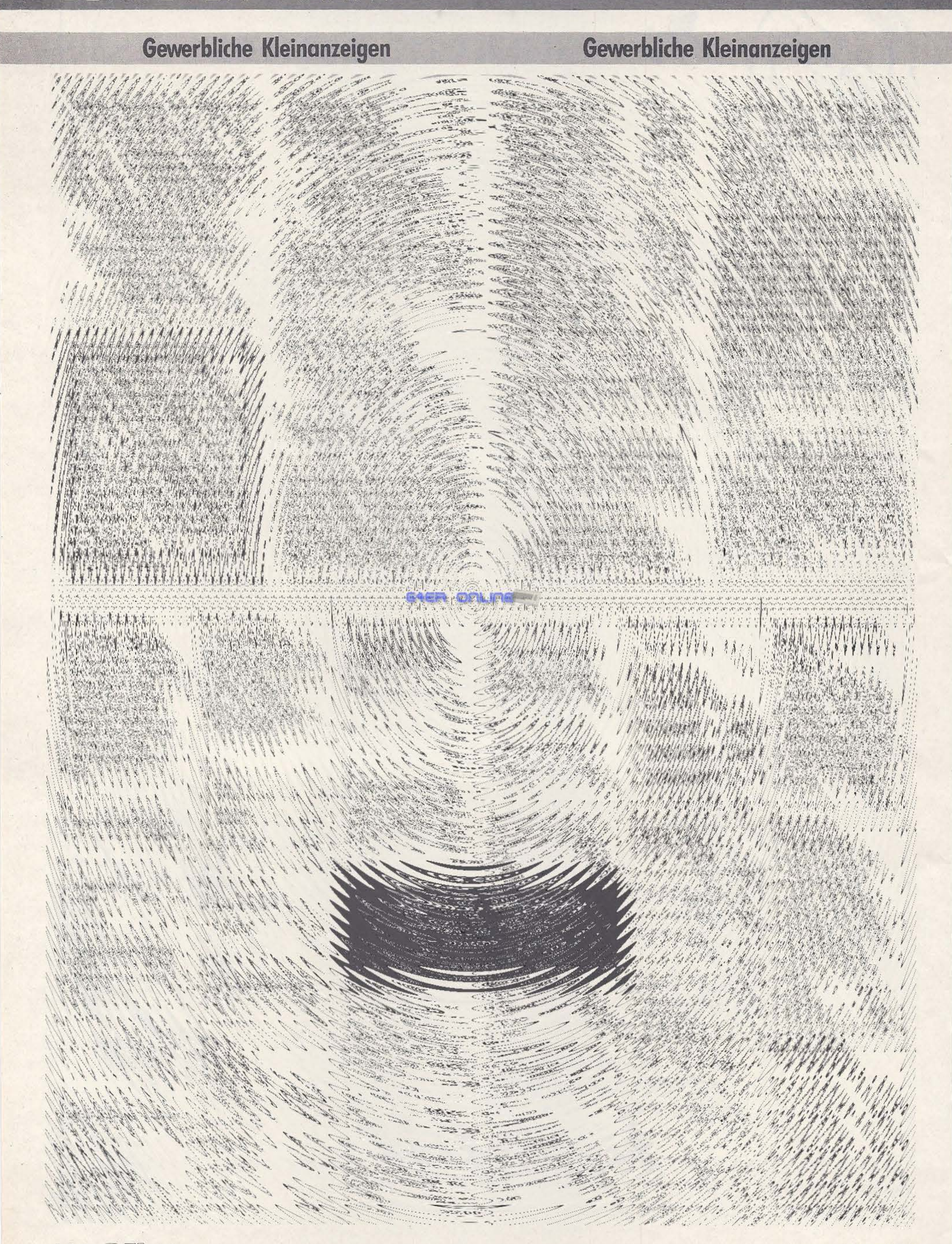
Gewerbliche Kleinanzeigen





Gewerbliche Kleinanzeigen

Gewerbliche Kleinanzeigen



Grafikbücher zum C 64

Eines der faszinierendsten Dinge bei der Arbeit mit dem C 64 sind dessen Grafikfähigkeiten. Da jedoch die Programmierung nicht ganz einfach ist, nehmen sich viele Bücher dieses Themas an. Einige davon möchten wir heute vorstellen, wobei auffällt, daß jedes Buch die Grafik unter einem anderen Aspekt behandelt. Verwirrend sind in diesem Zusammenhang die vielen, ähnlich klingenden Titel der Bücher.

Grafik für Anfänger

Wirklich empfehlenswert für den Anfänger ist das Buch »Grafik und Musik auf dem Commodore 64«. Es vermittelt Kenntnisse aus allen Bereichen der Grafikprogrammierung, die mit vielen in Basic geschriebenen Beispielen erklärt werden. Mit Hilfe eines Selbsttests am Ende jedes Kapitels kann man sein bisher gewonnenes Wissen überprüfen und feststellen, ob man gegebenenfalls etwas nachlesen muß.

Schwerpunkt der ersten vier Kapitel ist die Programmierung von Sprites, wobei alle notwendigen Schritte ausführlich erläutert werden. Aber auch die Umdefinierung des Zeichensatzes und die hochauflösende Grafik werden in einer auch für den Anfänger verständlichen Weise angesprochen. Der Schwerpunkt liegt aber, wie gesagt, in der Spriteprogrammierung.

So beschäftigt sich dann auch das letzte Grafikkapitel mit der Verknüpfung von Sprites und Hintergrund. Der Musikteil des Buches, auf den in diesem Zusammenhang allerdings nicht näher eingegangen werden soll, führt den Leser schrittweise an die SID-Programmierung heran.

Im Anhang findet man schließlich nochmals alle Registerbelegungen, die man im Zusammenhang mit Ton und Grafik benötigt, tabellarisch zusammengefaßt. Mehrere Entwurfsblätter für Sprites und Sonderzeichen, die man bei Bedarf abzeichnen oder herauskopieren kann, runden dieses Buch ab.

Info: Stan Krute, Grafik und Musik auf dem Commodore 64, Markt & Technik Verlag, 336 Seiten, ISBN 3-89090-033-X, Preis 38 Mark

Grafik für Spielefreaks

Wer mit Vorliebe Spiele programmiert, sollte sich dieses Buch anschaffen. Denn es beschreibt nicht nur die Programmierung von Sprites oder das Arbeiten mit hochauflösender Grafik, sondern geht auch auf alle im Zusammenhang mit Spielen notwendigen Dinge ein. Dazu gehört zum Beispiel ein Kapitel über die Optimierung von Basic-Programmen, die Besonderheiten des PRINT-Befehls und die Erzeugung von großen Überschriften und Titelbildern (gerade letzteres nimmt in diesem Buch großen Raum ein). Auch hier werden alle Themen durch ausführliche Beispiele verdeutlicht. Im fast 100seitigen Anhang (!) finden sich dann noch weitere Listings, mit denen man Zeichen und Sprites oder überdimensionale Programmtitel editieren kann. Aber auch bereits fertige Zeichensätze sind dort in Form von ausgefüllten Entwurfsblättern zu finden.

Im Vergleich zu anderen Grafikbüchern wird hier mehr Wert auf die Beschreibung der Spielprogrammierung gelegt, was nicht unbedingt negativ zu beurteilen ist.

Info: Charles Platt, Commodore 64 Grafik und Design, Sybex Verlag, 280 Seiten, ISBN 3-88745-073-6, Preis: 39 Mark

Bücher zu Basic-Erweiterungen

Die nächsten beiden Bücher setzen eine Basic-Erweiterung voraus. Daher sollte man beim Kauf darauf achten, ob man eine dieser Erweiterungen besitzt.

»C 64 Akustik und Grafik« heißt ein Buch aus dem te-wi-Verlag. Ein Teil (Kapitel 5 bis 7) setzt entweder die Erweiterung Simons- oder Structured-Basic voraus.

Nachdem der Wert eines Floppy-Laufwerkes klargemacht wurde, beginnt der Autor im vierten Kapitel mit der Beschreibung der Grafikzeichen auf der Tastatur, ein Thema, mit dem sich blutige Anfänger befassen.

Die eigentliche Beschreibung der Befehle des erweiterten Basic schließt sich daran an. Dabei beschränkt sich das Buch nicht nur auf die Grafikbefehle, sondern geht auf die gesamte Erweiterung ein. Diese Ausführungen sind sehr übersichtlich gehalten und mit jeweils einem oder mehreren Beispielen versehen worden. Sämtliche Beschreibungen finden zweigleisig statt, das heißt ein Befehl des Simons- wird dem des Structured-Basic (eine Erweiterung aus dem Hause te-wi) gegenübergestellt.

Auch auf die Definition und Handhabung der Sprites durch Simons-Basic wird ausführlich eingegangen.

Das Gesamturteil fällt hier sehr gemischt aus, denn vom Niveau steht der erste Teil des Buches in keinem Verhältnis zum zweiten. Dies ist ein Manko, mit dem allerdings nicht nur dieses Buch behaftet ist.

Info: John J. Anderson, C 64 Akustik und Grafik, te-wi Verlag, 190 Seiten, ISBN 3-921803-31-4, Preis 49 Mark

»Grafik für Mathematikinteressierte«

Auch dieses Werk ist für den Anwender von Simons-Basic gedacht. Es ist in seiner Art wesentlich anspruchsvoller als andere Bücher zu diesem Thema. Hier wird nämlich nicht auf die Programmiertricks zur Erzeugung einer Hires-Grafik eingegangen, sondern vielmehr auf die Benutzung des Computers für mathematische Zwecke. Daher sollte man auch etwas Interesse und Kenntnisse für dieses Gebiet mitbringen, denn die Algorithmen sind nicht immer einfach nachzuvollziehen.

Wer sich dennoch mit diesem Buch befaßt, wird mit teilweise sehr schönen Hires-Grafiken belohnt. Zunächst einmal beschreibt der Autor die einzelnen Grafikbefehle und zeigt deren Anwendungsgebiete anhand vieler Beispiele auf. Weiterhin sind viele Demografiken vorhanden, wie zum Beispiel Moiree-Effekt oder die verschiedensten Röhrengrafiken. Aber auch das Zeichnen verschiedener Funktionen ist Thema dieses Buches.

Behandelt werden weiterhin:
— Präsentationsgrafiken (Balken- und Kuchendiagramme)
— Zufallsgrafiken
— Verschiedene Interpolationsalgorithmen.

Sehr interessant ist ferner die Abhandlung über dreidimensionale Darstellungen, wobei es auch hier nicht ohne Theorie geht. Sowohl die Erzeugung von Grafiken ohne als auch mit verdeckten Linien sind Thema dieses Kapitels. Auch hierzu sind wieder viele Beispielprogramme abgedruckt, mit denen man die kuriosesten 3-D-Funktionen zeichnen kann. Schließlich werden ausgewählte Anwendungen angesprochen, die alle auf Simons-Basic aufbauen.

Fazit: Dieses Buch aus dem Westermann Schulbuchverlag ist zwar etwas anspruchsvoller als die anderen, dennoch ist es nicht nur für Mathematiker lesenswert.

Info: Walter Bachmann, Grafik auf C 64, Westermann Schulbuchverlag, 204 Seiten, ISBN 3-14-508811-4, Preis 49 Mark

Grafik für Profis

Gerade komplexere Programme im Zusammenhang mit Grafik sind in Basic sehr langsam. Da bleibt dann oft nur noch der Wechsel der Programmiersprache, also statt Basic »Grafik in Maschinensprache auf dem Commodore 64«.

Zunächst einmal muß man das Problem »wo speichere ich was« in den Griff bekommen. Das Buch gibt dabei in seinem zweiten Kapitel eine Hilfestellung, in dem es aufzeigt wie die einzelnen Register des Grafikchips programmiert werden müssen. Nach und nach werden nun alle Betriebsarten des Video-Chips angesprochen und anhand von Beispielen in Maschinensprache erläutert. Gleichzeitig wurden zum besseren Verständnis auch die analogen Basic-Programme abgedruckt.

Nachdem man nun erfahren hat, wo man seine Grafik im Speicher unterbringen kann, geht es weiter mit der Definition von Sonderzeichen, sowohl einfarbig als auch in Multicolor. Das nächste Kapitel wird von der hochauflösenden Grafik beherrscht. Eine größere Maschinenroutine, die das Setzen und Löschen von einzelnen Punkten erlaubt, bildet dabei die Grundlage. So bereiten auch kompliziertere Grafiken keine Schwierigkeiten mehr.

Aufgrund dieses ausführlichen Teils kommen die Sprites in diesem Buch etwas zu kurz. Der Autor beschränkt sich bei seinen Ausführungen auf die Beschreibung der Register. Interessant wird es dann noch einmal im 6. Kapitel, wo es um die Besonderheiten des Video-Chips geht. Dazu gehören unter anderem das Softscrolling, die Rasterregister und die Beschreibung des Rasterinterrupts. Ein kleiner Anhang, der jedoch nicht sehr übersichtlich ist, bildet den Abschluß.

Dieses Buch kann man allen empfehlen, die von Basic auf Maschinensprache umsteigen wollen, denn durch die Zweigleisigkeit (Maschinenprogramm und dazu die analoge Basic-Routine) wird das Umdenken bei der neuen Form der Programmierung geübt.

Info: Jürgen Hegner, Grafik in Maschinensprache auf dem Commodore 64, IWT-Verlag, 140 Seiten, ISBN 3-88322-051-5, Preis 38 Mark

Das Grafikbuch zum C 64

Dieses Data-Becker-Buch führt den Grafik-Interessierten von den Grundlagen bis zu komplexen Anwendungen wie Computer Aided Design (CAD). Der Leser erfährt alles über Sprites, Hires- und Multicolor-Grafik. Auch die Anwendungsseite kommt nicht zu kurz. 3D-Grafikeffekte werden ebenso behandelt wie die verschiedenen grafischen Darstellungsformen. Viele Beispiele und eine abgedruckte Grafik-Erweiterung zum Commodore-Basic machen das Buch für jeden an Computer-Grafik Interessierten zur empfehlenswerten Lektüre.

Info: Axel Plenge, Das Grafikbuch zum Commodore 64, Data Becker, 296 Seiten, 39 Mark.

(Christoph Sauer/ev)

Flottes Türmchen

Das MEA-Interface von Kühn macht den C 64 zum Steuercomputer für vielseitige Anwendungen. Beispielsweise zur Überwachung von Maschinen oder zum Regeln einer Heizung, abhängig von der Außentemperatur.

Das MEA-Interfacesystem von Manfred Kühn zeigt, daß ein Schaltinterface nicht immer in einem mausgrauen Kästchen verschwinden muß. Die Platinen für die universell verwendbaren 8-Bit Schaltinterfaces können nämlich zu einem hübschen kleinen »Elektronik-Türmchen« gestapelt und verschraubt werden. Damit die Platine den Umwelteinflüssen nicht ganz hilflos ausgesetzt ist, wird der Platinenturm von oben und unten durch eine Plexiglasplatte geschützt. Dies ist der erste Eindruck vom äußeren Erscheinungsbild, des alles in allem sehr sauber aufgebauten MEA-Interface.

Die Verbindung mit dem VC 20 oder C 64 erfolgt über ein, ebenfalls in Plexiglas gehülltes, User-Port-Interface (I1-C 64). Dieses wird über zwei Flachbandkabel mit den eigentlichen 8-Bit-Ein- und Ausgabemodulen verbunden. Zur Auswahl stehen dabei drei verschiedene Typen, die je nach Anwendungsgebiet individuell zusammengestellt werden können:

Eingangsmodul (E8-1): Mit Hilfe dieses Bausteins, von dem bis zu vier Stück angeschlossen werden können, ist es möglich, Signale von außen aufzunehmen und im Computer zu verarbeiten. Die Eingänge dieses Moduls können sehr einfach über PEEK im Basic-Steuerprogramm abgefragt werden. Die Zustände der einzelnen Eingänge werden über Leuchtdioden angezeigt.

Die Eingangsmodule können mit einer Schaltbrücke auf verschiedene Geräteadressen eingestellt werden. Das ist sehr wichtig, wenn mehrere Eingangsmodule angeschlossen werden sollen.

Da die Spannungsversorgung des Computers einen solchen Stromverbrauch nicht verkraften kann, ist ein zusätzliches Netzgerät erforderlich. Es reicht aus, wenn dazu an irgendein Eingangsmodul eine Spannung im Bereich von 7 bis 28 Volt angelegt wird, da alle Module über eine durchgeschleifte Spannungsversorgung miteinander verbunden sind.

Die Schaltspannungen der Eingänge betragen etwa die Hälfte der Betriebsspannung.

Ausgangsmodul (A8-1): Dieser Baustein ist das Gegenstück zum eben beschriebenen Eingangsmodul. Mit dem Ausgangsmodul können Spannungen bis zu 40 Volt/0,5 Ampere Gleichstrom ohne Relais geschaltet werden. Die Ausgänge werden durch POKEn einer »1« in die entsprechende Adresse eingeschaltet. Am Interface wird jeder durchgeschaltete Ausgang mit einer leuchtenden LED signalisiert.

Von den Ausgangsmodulen A8-1 dürfen maximal zwei Stück angeschlossen werden. Die Ausgangsmodule werden getrennt von den Eingangsmodulen mit Strom versorgt. Beide haben aber über die Minusleitung miteinander Kontakt.

Eine sehr positive Eigenschaft dieses Ausgangsbausteins ist die sogenannte Zyklusüberwachung: Eine Timerschaltung überprüft, ob das Interface regelmäßig abgefragt wird. Sollte dies nicht der Fall sein (wenn das Steuerprogramm beispielsweise angehalten wurde oder abgestürzt ist), dann werden sämtliche Ausgänge nach einer kurzen Wartezeit abgeschaltet. Sollte sich diese Zusatzfunktion als störend erweisen, kann sie über eine steckbare Kontaktbrücke abgestellt werden.

Ausgangsmodul (AR4-1): Diese dritte und letzte Platine im MEA-System ist mit vier Relais bestückt. Es können Spannungen von 50 bis 250 Volt über vier Relais geschaltet werden. Dieses Ausgangsmodul sollte allerdings nur demjenigen vorbehalten sein, der sich mit Starkstrom auskennt. Alle 220 V-Verbindungen erfolgen über normale Steckdosen und Stecker. Als Schaltspannung reicht eine Spannung um 5 V.

Ansonsten sind die Funktionen des AR4-1 mit der A8-1-Platine identisch. Es ist auch hier eine Zyklusüberwachung vorhanden. Das Ausgangsmodul besitzt vier Schaltausgänge mit je 1250 Watt Schaltleistung.

Es können bis zu vier solche Module an das User-Port-Modul angeschlossen werden.

Programme und Dokumentation: Die Dokumentation ist, mit etwa 20 Seiten, ausreichend. Die mitgelieferte Software besteht im wesentlichen aus einem Maschinenprogramm, das die Verbindung zum Interface herstellt und einem Basic-Steuerprogramm, das die Grundlage für eigene Programmentwürfe bilden soll.

Fazit: Das durch eine besondere Bauweise auffallende MEA-Interfacesystem eignet sich unseres Erachtens besonders für kleine Meß- und Steueraufgaben oder Regelkreise. Passend zum MEA-Digitalsystem gibt es auch Analogmodule.

(Christoph Sauer/hm)

Bezugsquelle: Ingenieurbüro für Mikroelektronik-Anwendung Manfred Kühn, Friedrich-Ebert-Allee 61, 2000 Schenefeld, Tel. (040)8308738

Preise: User-Port-Interface (I1-C 64): 133 Mark, Eingangsmodul (E8-1): 94,50 Mark, Ausgangsmodul (A8-1): 122 Mark, Systempaket (2x E8-1, 2x A8-1 und I1-C 64: 598 Mark).



Brother TC-600: Das Multitalent

Wie der Verwandlungskünstler im Varieté, so versteht es auch der TC-600 sich zu wandeln: Er ist gleichzeitig Schreibmaschine, Drucker, Terminal und Computer. Wir haben getestet, was er am besten kann.

Bei soviel Auswahl fällt es nicht leicht, die wesentlichste Funktion des TC-600 (Bild 1) herauszustellen. Aber da er mit einem ausgezeichneten Thermo-Druckwerk (24 x 18 Punkte) ausgestattet ist, ist der TC-600 in erster Linie ein Drucker. Sowohl auf Normalpapier (mit Farbband) als auch auf Thermopapier liefert der TC-600 eine gestochen scharfe Schrift. An den C 64 wird der TC-600 über eine RS232-Schnittstelle angeschlossen. Durch den Batteriebetrieb (Netzanschluß ist auch möglich) bietet der handliche Drucker dem Besitzer eines C 64 wertvolle Dienste als mobiles Texterfassungssystem. Durch die sinnvolle Konzeption der bidirektionalen Schnittstelle ist es mit dem TC-600 tatsächlich leicht, Texte, die außer Haus erfaßt wurden, in den Speicher des C 64 zu übertragen. Der relativ groß dimensionierte Speicher von 14300 Zeichen erlaubt es nämlich, dem TC-600 Texte mit einer Länge von bis zu vier Schreibmaschinenseiten zu speichern.



Bild 1. Der TC-600 — ein Alleskönner

Das Schriftbild des Brother TC-600
Der Brother TC-600 ist ein Schreibmaschinen-Terminal-Textverarbeitun[illegible]-Drucker mit eigenem 24X18 Display, 14,3 KByte Textspeicher und umfangreichen Fähigkeiten.

Bild 2. Der Thermo-Druckkopf arbeitet mit Thermo- oder Normalpapier

Anschluß über RS232-Schnittstelle

Da der TC-600 ein Datenendgerät ist, muß er an den C 64 anders als beispielsweise ein Akustikkoppler angeschlossen werden. Zwischen der notwendigen RS232-Schnittstelle und dem TC-600 muß deshalb ein Null-Modem geschaltet werden. Die Einstellung der Parameter ist nicht ganz einfach. Bei 300 Baud lautet das optimale Protokoll 4BN8X1 beim TC-600 und OPEN 2,2,3,CHR$ (6+32)+CHR$ (32+128) beim C 64. Danach kann der Drucker mit den üblichen Befehlen wie CMD und PRINT# angesprochen werden. Leider sehen nicht alle Textverarbeitungsprogramme den Betrieb eines Druckers mit RS232-Schnittstelle vor. Hier heißt es erst prüfen, dann kaufen.

Allerdings ist der TC-600 so gut ausgestattet, daß er schon fast keinen Computer zur Texteingabe mehr benötigt. Mit seinem eingebauten einzeiligen Display für 24 Zeichen hat man sogar einen gewissen Überblick über das schon Geschriebene. Verläßt man den normalen Schreibmodus, so werden die Zeichen nicht mehr unmittelbar gedruckt, sondern erst nachdem eine ganze Zeile eingegeben ist. Bis dahin hat man Zeit, den Text beliebig oft zu verändern beziehungsweise Worte und Buchstaben zu korrigieren.

Eingebautes Terminal

Die Schrift (16 Zeichen pro Sekunde) hat eindeutige Briefqualität (Bild 2). Einziger Nachteil des Schriftbildes ist es, daß außer dem Unterstreichen praktisch keine Veränderungen an den Buchstaben mehr vorgenommen werden können. Hier fehlt die Fett- oder Breitschrift, ebenso wäre auch eine Grafikfähigkeit sicherlich machbar gewesen.

Neben dem Schreib- und dem Textverarbeitungsmodus verfügt der TC-600 noch über eine Terminalfunktion. Sie eröffnet ein weites Feld der Anwendungen. Erlaubt sie doch das Senden und Empfangen von Daten über einen Akustikkoppler. Eine Datenübermittlung aus der Telefonzelle zu anderen Computern ist also keineswegs Utopie.

Fazit

Allen denjenigen, die nicht nur zu Hause Texte erfassen möchten, sondern beispielsweise auch im Flugzeug oder Bahn, wird der TC-600 gute Dienste leisten. Als einziger Drucker am C 64 eignet er sich allerdings nur eingeschränkt. Die nicht vorhandene Grafikfähigkeit und das Fehlen der C 64-spezifischen Zeichen schränken die Verwendbarkeit als universellen Drucker ein.

Mit einem Preis von beinahe 1400 Mark ist der TC-600 für jemanden, der nur einen Drucker braucht, zu teuer. Wollen Sie auf Reisen Texte schreiben und zu Hause mit dem Computer weiterverarbeiten, ist der TC-600 ein guter Diener.

(Arnd Wängler/hm)

Info: Brother, Im Rosengarten 14, Postfach 1320, 6368 Bad Vilbel



Darf es etwas mehr sein?

Steckmodule verbrauchen fast immer Speicherplatz. Business Basic erweitert ihn auf 61 KByte RAM und bietet zudem über 50 neue Befehle.

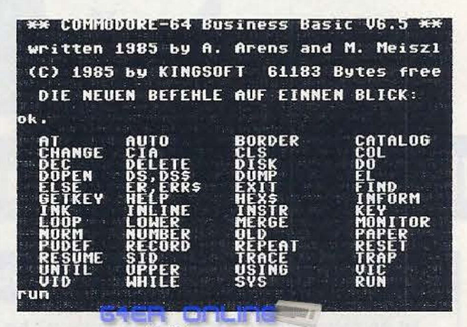
Der Prozessor des C 64 (MOS 6510) ist als 8-Bit-CPU in der Lage, genau 65536 Speicherzellen zu adressieren. Beim C 64 sind, im Gegensatz zum VC 20, alle diese Speicherzellen mit RAM-Bausteinen bestückt. Trotzdem bleiben nur 38911 Byte zum Programmieren in Basic übrig, denn 24 KByte ROM, 2 KByte für Zeropage und Bildschirmspeicher fordern ihren Tribut. Der Rest ist nur für Programme in Maschinensprache zugänglich. Diesem Manko hilft Business Basic von Kingsoft gründlich ab. Es ist schon beeindruckend, wenn man in der neuen Einschaltmeldung »61183 Bytes free« (Bild) lesen und über dieses Speichervolumen in Basic auch frei verfügen kann.

Insgesamt verfügt der C 64 mit Business Basic über 96 KByte (64 KByte RAM + 28 KByte ROM). Das Geheimnis, das hinter diesem Konzept steckt, ist die konsequente Weiterführung eines schon im Standard-C-64 vorhandenen Prinzips. Es wird, je nachdem welche Funktion benötigt wird, zwischen verschiedenen Speicherbereichen, ROM und RAM umgeschaltet. Für den Computer ist also immer nur der Teil des Speichers »sichtbar«, den er gerade benötigt. Dieses sogenannte »banking« hat aber auch seine Nachteile, denn es treten Geschwindigkeitseinbußen von zirka 18 bis 25 Prozent auf. Dafür wurden aber einige leistungsfähige Befehle und Programmiertricks implementiert, die diesen Nachteil relativieren. So dauert die gefürchtete Garbage Collection jetzt maximal 1,5 Sekunden, statt wie bisher bis zu 20 Minuten. Man kann jetzt also ohne weiteres mit sehr großen Stringfeldern arbeiten.

Basic-Befehle wie beim C 16

Wem die Ablaufgeschwindigkeit der unter Business Basic geschriebenen Programme trotz der Beseitigung der Garbage Collection zu gering ist, kann sie mit einem gleichfalls von Kingsoft angebotenen Spezialcompiler noch um einiges vervielfachen.

Das ist aber nur ein Teil der Vorteile, die das Modul zur Verfügung stellt. Denn neben den genannten Leistungen wird noch eine Basic-Erweiterung geboten, die allerdings speziell die Programmierung von Anwenderprogrammen unterstützt (daher auch der Name des Moduls). Bei der Namenswahl der



```
** COMMODORE-64 Business Basic V6.5 **
written 1985 by A. Arens and M. Meiszl
(C) 1985 by KINGSOFT  61183 Bytes free
 DIE NEUEN BEFEHLE AUF EINNEN BLICK:
ok.
  AT        AUTO       BORDER     CATALOG
  CHANGE    CIA        CLS        COL
  DEC       DELETE     DISK       DO
  DOPEN     DS,DS$     DUMP       EL
  ELSE      ER,ERR$    EXIT       FIND
  GETKEY    HELP       HEX$       INFORM
  INK       INLINE     INSTR      KEY
  LOOP      LOWER      MERGE      MONITOR
  NORM      NUMBER     OLD        PAPER
  PUDEF     RECORD     REPEAT     RESET
  RESUME    SID        TRACE      TRAP
  UNTIL     UPPER      USING      VIC
  VID       WHILE      SYS        RUN
run
```

Die neuen Befehle

neuen, zusätzlichen Befehle (Bild) wurde darauf geachtet, daß die Syntax und die Tokencodierung der des Basic 3.5 entspricht. Die Programme des C 16 und Plus/4 (beide mit Basic 3.5 ausgestattet) funktionieren aber nur sehr eingeschränkt. Eine Anpassung dürfte aber nicht allzu schwer sein.

Zu den zusätzlichen Befehlen von Business Basic gehört zunächst eine sinnvolle Erweiterung der Fähigkeiten des Basic-Editors. Es gibt Befehle wie AUTO zur automatischen Zeilennummervorgabe, DUMP zum Anzeigen der Variableninhalte, NUMBER zum Umnumerieren eines Programmes mit Anpassung aller GOTO- und GOSUB-Befehle, KEY zur Funktionstastenbelegung und FIND zum Auffinden bestimmter Programmteile. Interessant ist der CHANGE-Befehl, der ähnlich der FIND-Anweisung bestimmte Stellen des Programmes sucht und gegen neue Ausdrücke ersetzt.

Bei Fehlern im Programmablauf wird die fehlerhafte Zeile angezeigt und der Cursor an die betreffende Stelle gesetzt. Diese Eigenschaften lassen sich durch HELP ON/OFF ein- beziehungsweise ausschalten. Es gibt auch den Befehl TRACE, mit dem man den Programmablauf anhand der ausgedruckten Zeilennummern verfolgen kann. Dieser Befehl läßt sich auch programmieren, so daß man auch Teile des Programmes gezielt beobachten kann. Das ist besonders dann wichtig, wenn eine lange Leseschleife am Anfang des Programms steht. Außerdem kann man jetzt, wie bei Exbasic Level II, den Ausdruck des Listings auf dem Bildschirm mit den Cursortasten vor- und rückwärts steuern. Ferner ist der LIST-Befehl ebenfalls programmierbar, so daß kein Abbruch mehr nach der Ausführung dieses Befehls unter Programmkontrolle erfolgt.

Formatierte Ausgabe

Sehr nützlich bei der Arbeit mit Bildschirmmasken sind die Möglichkeiten zur formatierten Ausgabe von Zahlen und Zeichenketten mit PRINT USING, das auch auf dem Drucker funktioniert. Erwähnenswert ist hierbei die Möglichkeit mit PUDEF die Verwendung von Kommas und Punkten bei der formatierten Ausgabe von Zahlen an deutsche Verhältnisse anzugleichen (Dezimalkomma und Tausenderpunkt). Weiterhin kann man den Ort der Bildschirmausgabe mit PRINT AT und der Angabe der gewünschten Zeile und Spalte bestimmen. Diese gesteuerte Ausgabe läßt sich auch in Zeichenketten einbauen, indem man den Zeichencode, gefolgt von der entsprechenden Zeichencodierung für Zeile und Spalte, in einen String einbaut (CHR$(1) + CHR$(Zeile) + CHR$(Spalte)).

Auch Kommandos zur gesteuerten Eingabe von Zeichen fehlen nicht. Der Befehl INLINE etwa gestattet die Eingabe aller Zeichen (also auch Kommas und Doppelpunkte), während der INFORM-Befehl die Cursortasten bei der Eingabe blockiert. Ferner gibt es noch den Befehl GETKEY, der auf ein Zeichen von der Tastatur wartet und die lästige Abfrage mit einer IF-THEN-Schleife erspart.

Selbstverständlich wurden einige Befehle zur Diskettensteuerung eingebaut. Da gibt es zum Beispiel CATALOG zum Anzeigen eines Inhaltsverzeichnisses oder DISK zum Senden von Floppy-Kommandos. Der Fehlerkanal kann über die Variablen DS und DS$ abgefragt werden. Weiterhin wird die Verwendung relativer Dateien mit Befehlen wie RECORD und DOPEN unterstützt. Die Befehle LOAD, SAVE und VERIFY beziehen sich ohne Parameteranga-

be immer auf die Floppy, wobei man in einem mit CATALOG gelisteten Inhaltsverzeichnis vor das zu ladende Programm nur noch LOAD schreiben muß und kein Doppelpunkt mehr hinter den Namen notwendig wird. Auch wird jetzt mit der RUN-Taste das erste Programm von Diskette geladen und gestartet. Man kann aber auch jedes beliebige Programm von Diskette laden und starten, indem man RUN und dahinter den Namen des Programmes eintippt. Sehr interessant ist der Befehl MERGE, der nicht nur ein weiteres Programm an das im Speicher vorhandene anhängt, sondern zeilenweise einsortiert.

Auch die Anhänger der Kassettenspeicherung wurden nicht vergessen. Die Aufzeichnungsgeschwindigkeit auf Band wurde um ein Vielfaches erhöht. Trotzdem können Programme im normalen Aufzeichnungsformat geladen werden.

Zur übersichtlichen Programmierung wurden sowohl bestehende Befehle erweitert, als auch neue definiert. Das altbekannte IF-THEN wurde um die Möglichkeit der Verwendung eines ELSE-Falles bereichert. Auch kann man den Befehl RESTORE in Zusammenhang mit Zeilennummern benutzen. Ferner liefert die Abfrage ASC(CHR$(0)) keinen Fehler mehr, sondern den Wert Null.

Neu hinzugekommen sind Schleifenbefehle wie DO-LOOP-WHILE,

Fortsetzung auf Seite 132

Mathemat nicht im Test

An dieser Stelle sollte eigentlich ein Testbericht des Mathemat von Data Becker stehen. Wir ließen dieses an sich sehr interessante Programm von zwei Lehrern unabhängig voneinander testen. Beide kamen jedoch zu dem Urteil »Nicht empfehlenswert«. Zu viele Details sind unausgereift, zu viele Schwachstellen müßten noch bearbeitet werden. Selbst im Vorwort zum Mathemat liest man: »... Sollte also Ihr Examen, der Nobelpreis oder eine komplette Autobahnbrücke vom Rechenergebnis abhängen, so empfehlen wir, mit herkömmlichen Methoden noch einmal das Ergebnis zu kontrollieren.«

Aufgrund der vorliegenden Ergebnisse verzichten wir deshalb auf einen ausführlichen Bericht.

Was leistet Pilot?

Pilot ist eine wenig bekannte Programmiersprache, die speziell für Lehrzwecke entworfen wurde.

Für den C 64 sind mittlerweile zahlreiche Sprachen für jeden Geschmack erhältlich. Von Pascal über C bis zu Forth sind heute fast alle Computer-Idiome in einer auf dem C 64 lauffähigen Version vorhanden. Eine der weniger bekannten Sprachen ist Pilot, die für den C 64 von Commodore selbst angeboten wird.

Pilot ist eine sogenannte Interpretersprache. Sie verfügt über einen »Immediate-Mode« in dem alle eingegebenen Befehle sofort ausgeführt werden, einen »Edit-Mode« zur Erstellung von Programmen, den »Run-Mode« zur Benutzung der erstellten oder geladenen Programme, sowie den »Command-Mode«, von dem aus Disketten- und Druckeroperationen gehandhabt werden. Einmal von der mitgelieferten Diskette geladen, unternimmt man unter der Anleitung der — leider in Englisch verfaßten — Dokumentation die ersten Versuche, sich mit der neuen Sprache vertraut zu machen.

Bald schon wird klar, daß einer der Schwerpunkte von Pilot die Grafik ist. Mittels verschiedener Befehle kann man Punkte setzen, Linien ziehen und Flächen ausfüllen. Der Ausgangspunkt der Koordinaten, ursprünglich in der linken unteren Bildschirmecke angesiedelt, ist dabei nach Belieben veränderbar. Der Bildschirm ist durch den Split-Befehl in bis zu fünf Fenster unterteilbar, durch die Eingabe eines einfachen Programmes können Sprites erstellt und die vorgegebenen Buchstaben der einzelnen Tasten verändert werden. Dazu gibt man das neue Zeichen oder das Sprite in Punkten und »x« ein, wobei die Punkte für Hintergrundfarbe, die »x« für hervorgehobene »Pixels« stehen, wie das auch von anderen Sprite-Editoren her bekannt ist. Das fertige Sprite kann dann auf dem Bildschirm bewegt werden und mit dem Rand oder anderen Sprites kollidieren. Auch zur Einstellung der verschiedenen Sound-Parameter hält Pilot eine Reihe von Befehlen bereit, mit denen sich eine Tonkurve genau vordefinieren läßt.

Die bisher doch recht konventionell geratene Sprache hält bei der Stringbehandlung einige Überraschungen bereit. Der Befehl »Match« ermöglicht beispielsweise das Durchsuchen eines Antwortsatzes nach einem bestimmten Begriff. Dabei läßt sich auch die Möglichkeit berücksichtigen, daß das eingegebene Wort vom Programm-Anwender falsch geschrieben werden kann: Modifiziert man den »Match«-Befehl, so akzeptiert der Computer auch ein als »commadore« geschriebenes »commodore«. Der »Jump«-Befehl nimmt, statt der in Pilot nicht vorhandenen Zeilennummern, Namen als Sprungziele. Diese Möglichkeiten erlauben die einfache Programmierung von Vokabel-Trainingsprogrammen, Grammatikprogrammen oder auch Text-Adventures. Der interessanteste Befehl ist aber sicherlich »Execute«. Dabei wird die Antwort des Benutzers als Pilot-Befehl behandelt und erlaubt so die Veränderung eines Programmes, während es läuft. Dies ist beispielsweise bei der Eingabe von mathematischen Funktionen von Vorteil, was in Basic nicht so ohne weiteres zu bewerkstelligen ist. Eines der Haupthindernisse für eine umfangreichere oder professionellere Nutzung von Pilot ist, wie auch in vielen anderen Sprachen, der zur Verfügung stehende Speicherplatz, der mit 12 KByte nicht gerade exzessiv dimensioniert ist. Zwar kann mit dem »Link«-Befehl vom Programm aus ein weiteres Programm nachgeladen werden, aber diese Möglichkeit kann sicher nicht alle Speicherprobleme lösen, da das vorhergehende Programm dabei natürlich aus dem Arbeitsspeicher geworfen wird. Auch der für Maschinenroutinen zur Verfügung stehende Bereich, der 1024 Byte umfaßt, wird für größere Anwendungen sicher nicht genügen. So wird Pilot, obschon es eine Sprache mit einigen interessanten Details ist, die teilweise recht interessante Programmiermöglichkeiten eröffnen, doch wohl eher eine Exotenrolle beschieden bleiben. Aufgrund der Einfachheit der Sprache dürfte sie aber für Anfänger, die sich den Zugriff zu den Grafik-Möglichkeiten ihres C 64 erleichtern wollen, einer Überlegung wert sein.

(Christof Bachmair/ev)

Info: Pilot gibt es nur auf Diskette für den C 64 bei Ihrem Commodore-Händler. Preis: 59 Mark

Pascal für Profis

Das neue Profi-Pascal von Data Becker enthält alle Funktionen von Standard-Pascal und darüber hinaus viele Erweiterungen. Ein weiterer Pluspunkt: Der Diskettenzugriff wurde stark beschleunigt.

Im zweiten Anlauf hat Data Becker es geschafft: Nachdem das inzwischen schon etwas bejahrte Pascal 64 in verschiedenen Punkten nicht ganz zu überzeugen vermochte, liegt mit »Profi-Pascal« ein Compiler vor, den man jedem Pascal-Freund uneingeschränkt empfehlen kann. Neben dem von Kathleen Jensen und Niklaus Wirth im »Pascal User Manual and Report« beschriebenen Standard-Pascal, dessen Funktionen ohne Abstriche implementiert wurden, bietet Profi-Pascal viele sprachliche Erweiterungen, die Pascal-Freaks zum Teil aus UCSD-Pascal kennen.

Das Profi-Pascal-eigene Betriebssystem weist zwar einige kleine Schwächen auf, die jedoch ein riesiger Vorteil gegenüber anderen C 64-Programmen aufhebt: Der Zugriff auf Disketten erfolgt rund viermal schneller. Normalerweise werden 250 bis 400 Byte pro Sekunde von der Floppy in den Arbeitsspeicher (oder umgekehrt) übertragen. Unter Profi-Pascal sind es jedoch 1250 Byte pro Sekunde. Diese softwaremäßig erreichte Steigerung fällt äußerst positiv auf.

Das Hauptmenü (Bild 1) vermittelt einen Überblick über die wichtigsten Teile des Pascal-Systems. Compiler, Assembler, Editor und Betriebssystem sind auf einer Diskette gespeichert. Auf diese Diskette sollte der Programmierer achten wie auf seinen Augapfel. Sie läßt sich zwar kopieren, aber Compiler und Assembler sind dann nicht mehr lauffähig. Lediglich der Editor und die Utility-Dateien sind problemlos zu duplizieren. Pro Originaldiskette kann zusätzlich eine Sicherungskopie erworben werden. Wird die Diskette im Laufe der Zeit beschädigt oder abgenutzt, so kann sie für zwanzig Mark gegen eine Ersatzdiskette umgetauscht werden.

Obwohl Profi-Pascal mit zwei Laufwerken arbeiten kann, reicht ein Laufwerk bereits aus, um ohne häufiges Diskettenwechseln zu arbeiten, da beim Übersetzen die Quelldatei immer auf der Systemdiskette gespeichert sein muß.

Pascal mit vielen Extras

Jedes Pascal-Programm hat denselben blockorientierten Aufbau. Der Programmblock besteht aus einem Vereinbarungs- und einem Anweisungsteil. Der Vereinbarungsteil ist wiederum in die Definition von Labels, Konstanten, Typen, Variablen und Prozeduren unterteilt. Abweichend von Standard-Pascal dürfen bei Profi-Pascal in der Konstantenvereinbarung auch einfache Ausdrücke stehen, die wiederum Konstanten enthalten. Hierzu ein Beispiel:

```
Const faktor = 3.75;
e-wert = faktor * 10 + 5;
```

Die Liste der vordefinierten Konstanten wurde um Pi (= 3.1415926536) und STKPOI als Zeiger auf den Pascal-Variablen-Stack erweitert. Der Zeiger deutet auf die unterste Speicherstelle des von oben nach unten wachsenden Variablenstack.

Die Typen Integer, Real, Boolean und Char sind standardmäßig vordefiniert. Die größte ganze Zahl ist 32767. Der größte Real-Wert lautet ± 3.4028236692E+38 und der kleinste von Null unterscheidbare Wert beträgt ± 8.8162076312E-39. Array, Set, Record und File sind wie im Standard vorhanden. Sogar variante Records sind erlaubt. Das Attribut »PACKED« wird akzeptiert, hat jedoch keine Wirkung: Profi-Pascal speichert Daten von vornherein platzsparender, so daß Komprimieren von Werten überflüssig ist. In anderen Pascal-Versionen belegt beispielsweise eine Variable vom Typ »Char« 2 Byte, in Profi-Pascal hingegen nur 1 Byte.

Mit dem zusätzlichen String-Typ können Zeichenfolgen bis zur maximalen Länge von 132 Zeichen definiert werden. Zwischen Variablen der Typen »Array of Char« und String sind Wertzuweisungen sowie Vergleiche direkt, also ohne Indizierung möglich.

Profi-Pascal wurde um die System-Variablen »MEM« und RANDOM« erweitert. Die Variable MEM ist so definiert, daß der gesamte Speicher des Commodore 64 als Array-Variable angesprochen werden kann. Schreiben in oder Lesen von einer Speicherstelle ist damit möglich. MEM ersetzt PEEK und POKE vollständig. Die Variable RANDOM liefert bei jeder Zuweisung eine neue Zufallszahl.

Bei der Definition von Prozeduren und Funktionen ist besonders bemerkenswert, daß in der Parameterliste Prozeduren und Funktionen übergeben werden dürfen. Dies entspricht dem Standard, ist jedoch in kaum einer anderen Pascal-Version für Mikrocomputer vorgesehen. Man kann in Profi-Pascal also beispielsweise eine Prozedur schreiben, die Nullstellen mathematischer Funktionen berechnet. Die Funktion selbst kann dann einfach über die Parameterliste übergeben werden.

Relative Dateien möglich

Außerdem sieht Profi-Pascal vor, Dateivariablen zu indizieren. Damit erlaubt es den wahlfreien Zugriff auf ein einzelnes, genau definiertes Element einer Datei. Damit sind relative Dateien möglich geworden, was weit über den Pascal-Standard hinausgeht, der nur sequentielle Dateien vorsieht.

Der Zugriff auf Teilbereiche eines Arrays oder Strings ohne Indizierung (!) ist vor allem bei Arrays of Char oder Stringgrößen praktisch. Beispiel:

```
Var Feld:Array(1..10) of char;
Feld(>5):='Sonne';
```

An die Komponenten des Arrays mit dem Namen Feld wird ab der fünften Position das Wort »Sonne« übergeben. Es paßt genau in das Array. Der Programmierer muß unbedingt darauf achten, daß der zugewiesene Wert die Grenze des Arrays nicht überschreitet.

Ein Ärgernis für viele Pascal-Programmierer ist in Profi-Pascal behoben. Die Case-Anweisung wurde um einen Else-Zweig vervollständigt. Falls für bestimmte Labels in der Case-Anweisung kein Auswahlzweig vorgesehen ist, wird die dem Else folgende Anweisung ausgeführt. Beispiel:

```
Case note of
1:Writeln('super');
2:writeln('ganz gut')
else writeln('geht schon')
end;
```

Zu den vordefinierten Prozeduren und Funktionen sind in Profi-Pascal noch eine beachtliche Anzahl hinzu gekommen. Die wichtigsten werden nun kurz erwähnt.

Über die Prozedur Allocate bestimmt der Programmierer selbst, an welchem Speicherplatz der Wert einer Zeigervariablen abgelegt wird. Bei New legt das System selbst die Adresse fest. Man muß jedoch die Aufteilung des Speicherbereichs genau kennen, da man leicht in Gefahr gerät, den Programmcode oder andere Variablen zu überschreiben.

Programme verketten

Continue ruft ein neues Pascal-Programm aus dem laufenden heraus auf und startet es. Ein Rücksprung in das ursprüngliche Programm wird nicht ausgeführt. Die Prozedur Execute funktioniert ähnlich wie Continue, behandelt aber das aufgerufene Programm wie ein Unterprogramm und springt nach dessen Abarbeitung in das ursprüngliche Programm zurück.

Hex gibt Integer-Größen als hexadezimale Zahlen aus, und Induc schaltet auf ein anderes Eingabegerät um, wobei die Geräteadresse anzugeben ist. Dementsprechend lenkt Outdvc die Ausgabe auf ein anderes Gerät. Setdrv bestimmt das aktuelle Arbeitslaufwerk für den Diskettenzugriff. Die Prozeduren zur Dateiverwaltung Reset, Rewrite oder Seek greifen dann auf das mit Setdrv vereinbarte Laufwerk zu. Close und Lock schließen Dateien, die im Programm mit Reset oder Rewrite eröffnet wurden. Dateien, die mit Lock gesichert werden, sind gegen Überschreiben oder Löschen geschützt. Kill löscht Dateien vom Programm aus. Seek setzt den Zugriffszeiger auf eine Datei so, daß er auf einen bestimmten Satz weist. Dieser Satz kann dann mit Get gelesen werden.

Mark und Release steuern die Speicherplatzverwaltung auf dem Heap. Der Heap ist die Speicherplatzhalde für dynamische Variable. Nach dem Aufruf von Mark weist der Zeiger auf die oberste Position der von unten nach oben wachsenden Halde. Dann teilt die Prozedur New einer Variablen einen neuen Speicherplatz zu und die Halde ist weiter gewachsen. Release setzt den Haldenspeicherplatz wieder auf das Maß zurück, wie es mit Mark festgehalten wurde. Mark und Release ersetzen die Prozedur Dispose aus Standard-Pascal.

Die Funktion Anykey stellt fest, ob eine Taste gedrückt wurde. Getkey liefert das über die Tastatur eingegebene Zeichen. Ioerror gibt Auskunft darüber, ob bei einem Ein-/Ausgabeprozeß ein Fehler aufgetreten ist.

Overlay-Technik

Sehr lange Programme, die nicht mehr in den Arbeitsspeicher passen, können in Segmente zerlegt werden. In einem Programm dürfen maximal acht Segmente stehen. Der Compiler behandelt die Segmente so, als ob sie parallel im Speicher liegen würden. Er merkt sich die Anfangsadresse des ersten Segments und compiliert die folgenden jeweils für dieselbe Adresse. Der gesamte für die Segmente reservierte Arbeitsplatz richtet sich nach dem längsten Segment. Bei der Overlay-Technik muß der Programmierer bedenken, daß sich Segmente nicht gleichzeitig im Arbeitsspeicher befinden, sondern bei Bedarf von der Diskette nachgeladen werden und jeweils die Codedatei immer zur Verfügung stehen muß. Trotzdem ist es nicht ausgeschlossen, daß sich Segmente gegenseitig aufrufen. Eine Alternative zur Overlay-Technik stellen die Prozeduren Execute und Continue sowie auch externe Unterprogramme dar.

Externe Routinen

Externe Routinen können in Pascal oder Assembler geschrieben sein. Sie müssen vor ihrem Aufruf compiliert und an eine bestimmte Adresse im Arbeitsspeicher geladen werden. Dabei sind Speicherkollisionen zu vermeiden. An externe Unterprogramme können Werte über die Parameterliste übergeben werden.

Als weiterer Leckerbissen für den Profi können Assembler-Routinen direkt in Pascal-Programme einge-

Bild 1. Hauptmenü von Profi-Pascal und das Inhaltsverzeichnis der Systemdiskette. Systemdateien sind invers geschrieben.

Bild 2. Das »Statistical Summary« liefert Detailinformationen

Bild 3. Viele praktische Funktionen zum Bearbeiten von Dateien

fügt und gleich mitübersetzt werden. Die Anweisung Assemble in der Kopfzeile der Routine gibt dem System zu verstehen, daß eine eingebettete Maschinenroutine folgt, die mit dem integrierten Assembler zu übersetzen ist. Wie bei Pascal üblich, erzeugt der Compiler keinen reinen Maschinencode, sondern einen Zwischencode, den sogenannten P-Code, der vom Laufzeit-System während der Programmausführung interpretiert wird.

Ehe der Compiler beginnt, einen Quelltext in P-Code zu übersetzen, stellt er mehrere Fragen. Der Programmierer hat nun die Wahl, Compiler-Optionen einzustellen oder vorgegebene Werte zu übernehmen. Er kann bestimmen, an welcher Adresse die Halde und an welcher Position der Stack beginnt. Äußerst hilfreich bei der Suche nach Laufzeitfehlern ist der »Post Mortem Dump«. Sobald ein Fehler auftritt, informiert das System über die Programmstelle, an der der Fehler liegt. Man erfährt, zu welcher Prozedur oder Funktion die Zeile gehört und wie tief die Unterprogramme verschachtelt sind. Zusätzlich werden die Werte sämtlicher Variablen zum Zeitpunkt des Programmabbruchs ausgegeben. Außerdem druckt das System wahlweise den P-Code aus sowie auch ein Protokoll der Übersetzung.

Fehlermeldungen werden mit einer Nummer angezeigt. Der Programmierer muß dann im Handbuch nachlesen, um welchen Fehler es sich handelt. Nach dem Drücken der Leertaste setzt das System die Übersetzung fort. In den Editor verzweigt das System nach dem Tippen der Run/Stop-Taste. Nachdem das Programm vollständig übersetzt worden ist, stellt der Compiler noch eine kleine statistische Analyse auf (Bild 2). Sie enthält beispielsweise die Anzahl der Konstanten oder Variablen im Programm. Der Code wird automatisch unter dem Programmnamen, der in der Kopfzeile angegeben wurde, auf der Diskette gespeichert.

Zeilenorientierter Editor

Der Editor von Profi-Pascal arbeitet zeilenorientiert. Diese einfache Lösung bietet den Vorteil, daß er wenig Platz im Arbeitsspeicher belegt und für das Programm rund 43 KByte übrig bleiben. Nachteilig ist jedoch, daß beim Einfügen von Programmteilen große Sorgfalt auf die Numerierung der Zeilen verwandt werden muß. Eine Zeile faßt 80 Zeichen und ist mit RETURN abzuschließen. Der Editor ähnelt ansonsten stark dem des Basic-Systems.

Der Quelltext des Editors wird mitgeliefert. Er ist bis auf einige zeitkritische Prozeduren, die in Assembler programmmiert wurden, in Pascal geschrieben. So kann der Benutzer den Editor nach eigenen Vorstellungen verändern.

Profi-Pascal enthält zusätzlich noch einen Programmteil Utility (Bild 3) mit Funktionen zur Verwaltung von Dateien und Disketten. Dazu zählen das Initialisieren einer Diskette, Kopieren von Dateien und Disketten, Löschen, Laden und Umbenennen von Dateien sowie das Sichern einer Datei vor Überschreiben und die Ausgabe des Inhaltsverzeichnisses.

Beim Arbeiten mit dem Utility-Paket ist Vorsicht geboten, wie auch das Handbuch anmerkt. Auf falsche Eingaben wird teilweise äußerst garstig reagiert. Erst das Drücken des Netzschalters führt zurück ins System.

Das deutsche Handbuch ist sehr ausführlich gehalten, enthält ein Stichwortverzeichnis und ist 326 Seiten dick. Der Leser muß allerdings bereits mit Pascal vertraut sein, denn das Handbuch kann und soll kein Pascal-Lehrbuch ersetzen.

Trotz dieser kleinen Schwächen erfreut Profi-Pascal das Herz eines Pascal-Programmierers. Der Wirth-Standard wird erstmals bei einem C 64-Pascal ohne Abstriche erfüllt. Viele zusätzliche Erweiterungen ermöglichen das volle Ausnutzen aller Fähigkeiten des C 64-Systems, durch den integrierten Assembler sind sehr effiziente, maschinennahe Programme möglich. Relative Dateien verbessern die Datenverwaltung gegenüber Standard-Pascal enorm.

Alles in allem handelt es sich bei Profi-Pascal also um eine der leistungsfähigsten Pascal-Versionen überhaupt. Sowohl dem Pascal-Puristen als auch dem C 64-System-Spezialisten steht damit ein ideales Werkzeug zur Verfügung, das sich zu Recht mit dem Attribut »Profi« schmücken darf.

(Silvia Gutschmidt/ev)

Info: Profi-Pascal für Commodore 64, Data Becker, 198 Mark.

Positiv

— Profi-Pascal entspricht Standard-Pascal. Lediglich die Pointerfunktion Dispose wurde durch Mark und Release ersetzt.
— Schneller Diskettenzugriff: 1250 Byte pro Sekunde statt der üblichen 250 bis 400 Byte pro Sekunde.
— Zeilenorientierter Editor, der wenig Speicher belegt. Es bleiben rund 43 KByte zum Programmieren.
— Quelltext des Editors wird mitgeliefert. Er kann verändert werden.
— Kompakte Speicherung auf der Diskette,
— Zugriff über »Array of Byte« auf den gesamten Arbeitsspeicher. PEEK und POKE sind dadurch überflüssig.
— Compiler liefert Post Mortem Dump, der bei der Fehlersuche sehr nützlich ist.
— Assembler-Programme können in Pascal-Programme eingefügt und gleich mitübersetzt werden.
— Random-Zugriff auf Dateien,
— Segmentprozeduren,
— Verkettung von Programmen über Continue oder Execute.
— Viele zusätzliche, nützliche, vordefinierte Typen und Funktionen.
— Der erzeugte P-Code läuft verhältnismäßig schnell.
— Ausführliches Handbuch
— Backup- und Ersatzdisketten erhältlich

Negativ

— Systemdiskette nicht kopierbar.
— Beim Compilieren muß die Quelldatei auf der Systemdiskette gespeichert sein. Bei häufiger Benutzung führt dies zu einer raschen Abnutzung der Systemdiskette.
— Im Inhaltsverzeichnis sind Text- und Codedateien nicht gekennzeichnet.
— Das System bringt keine Warnung, wenn eine bereits existierende Datei überschrieben wird.
— Meldungen über Laufzeit- und Syntaxfehler sind teilweise schwer verständlich.
— Systemmeldungen in englisch.



Assembler ist keine Alchimie — Teil 11

Noch einmal geht es um Unterbrechungen. Wie geht unser Computer bei den verschiedenen Möglichkeiten vor? Wo kann man selbst einhaken und wie schreibt man eigene Unterbrechungsprogramme? Anhand von zwei Beispielen lernen Sie in dieser Folge, wie man den Ausstieg aus einem Programm verhindert und wie man die Rasterzeilenunterbrechung für eigene Programme einsetzt.

In der letzten Folge haben Sie erfahren, wie unsere CPU mit Unterbrechungen umgeht, welche Sorten von Unterbrechungen es gibt, welches Instrumentarium die Assembler-Sprache zur Behandlung dieser speziellen Technik bietet, und Sie kennen die »primären« Quellen für eine Unterbrechung. Diesmal wollen wir analysieren, wie die Unterbrechungen softwaremäßig bearbeitet werden, um Wege zu finden, die uns auf möglichst einfache Weise Eingriffe erlauben.

Der normale Verlauf eines IRQ

Neulich hatten wir bereits festgestellt, daß eine IRQ-Anforderung (nach dem Retten des Programmzählers und des Prozessorstatus-Registers, sowie dem Setzen der I-Flagge) den Inhalt des Vektors $FFFE/FFFF in den Programmzähler holt. Dort steht die Adresse $FF48 (dez. 65352) und deshalb startet nun das dort im ROM verankerte Programm, welches wir uns nun im einzelnen ansehen werden (alle Adressen als Dezimalzahlen, in Bild 1 finden Sie das Flußdiagramm dazu).

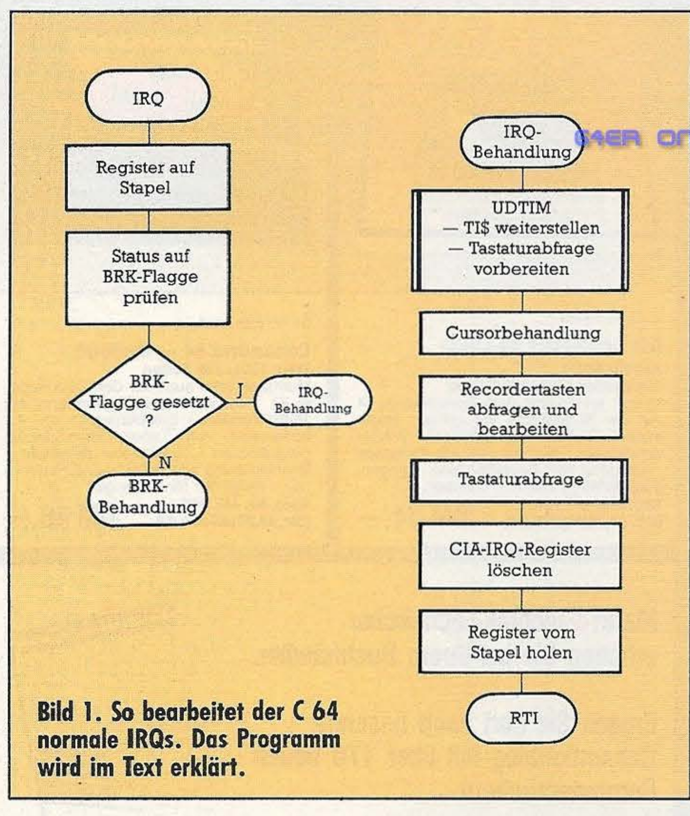


Bild 1. So bearbeitet der C 64 normale IRQs. Das Programm wird im Text erklärt.

65352	**PHA**	Zunächst werden der Akku und die
	TXA	Register X und Y auf den Stapel ge-
	PHA	schoben
	TYA	
	PHA	

Trickreich sind die beiden folgenden Befehle, mit denen das zu Beginn durch die CPU gerettete Statusregister gelesen wird:

	TSX	Stapelzeiger ins X- Register
	LDA 260,X	Einladen des Status-Registers

Nun wird geprüft, ob die BRK-Flagge gesetzt ist. Wenn das der Fall ist, dann ist der Auslöser ein BRK gewesen, ansonsten ein IRQ:

	AND #16	Isolieren der BRK-Flagge
	BEQ 65368	Wenn keine BRK-Flagge, dann überspringen des nächsten Befehls.
65365	**JMP (790)**	Falls BRK
65368	**JMP (788)**	Falls IRQ

Den vorletzten Sprungbefehl werden wir bei der BRK-Behandlung verfolgen. Interessant für uns ist jetzt der indirekte Sprung bei 65368. Der Vektor 788/789 ($314/315) liegt im RAM! Damit können wir ihn auf eigene Routinen verstellen. Genau hier ist der Ansatzpunkt für nahezu alle Eingriffe in die Unterbrechungsbehandlung. Der voreingestellte Wert in diesem Vektor ist die Adresse 59953 ($EA31). Das dort angesiedelte Programm wird im Normalfall 60mal in der Sekunde ausgeführt:

59953	**JSR 65514**	Das ist ein Kernal-Sprungbefehl zur Routine UDTIM bei 63131.

In diesem Unterprogramm wird zuerst die Uhr TI$ weitergestellt und dann die Tastaturabfrage vorbereitet.

59956 bis 60000		In diesem Programmteil erfolgt die Cursorbehandlung.
60001 bis 60026		Anschließend wird abgefragt, ob eine Recordertaste gedrückt ist und entsprechende Flaggen bearbeitet.
60027	**JSR 60039**	Dieses Unterprogramm dient zur Tastaturabfrage.

Auch in dieser Routine tritt übrigens ein indirekter Sprung nach einem RAM-Vektor auf (655/656 = $28F/290), der normalerweise auf 60232 zeigt, aber auch auf eine eigene Routine verbogen werden könnte.

Enthalten in der Tastaturabfrage ist auch die Überprüfung der RUN/STOP-Taste, die aber nur zusammen mit den in dem UDTIM-Aufruf voreingestellten Flaggen funktioniert. Deshalb wird das Abschalten der RUN/STOP-Taste im allgemeinen dadurch durchgeführt, daß man den IRQ-Vektor auf 59956 stellt und damit den ersten JSR-Befehl überspringt. Allerdings wird auf diese Weise auch die TI$-Uhr nicht weitergestellt.

60030	**LDA 56333**	Das ist das Unterbrechungs-Kontrollregister des IRQ-CIA, das hier durch Auslesen gelöscht wird.

Den Abschluß der IRQ-Routine bildet nun noch das Zurückschreiben der Register:

60033	**PLA**	Zurückholen des
	TAY	Y- und
	PLA	
	TAX	des X-Registers
	PLA	sowie des Akku.
60038	**RTI**	Damit kehrt der Computer zu dem durch den IRQ unterbrochenen Programm zurück.

Somit hätten wir's. Nun können wir je nach Bedarf entscheiden, welche von diesen Servicetätigkeiten wir bei einem eigenen IRQ-Programm brauchen: Die Uhr TI$, die Cursorbehandlung, die Abfrage der Recordertasten und die Tastaturabfrage.

Sehen wir uns nun an, was geschieht, wenn ein BRK-Kommando der Auslöser war.

BRK-Unterbrechung

Wir hatten vorhin am Scheideweg zwischen IRQ und BRK den letzteren links liegen gelassen. Normalerweise verwendet man beim Programmieren in Assembler ja ein Software-Instrument wie zum Beispiel den SMON, der so gebaut ist, daß der BRK-Vektor, den wir vorhin kennengelernt haben ($316/317 = 790/791) auf die Registeranzeige weist. Was geschieht eigentlich, wenn der BRK-Vektor unverändert bleibt, so also, wie er im Einschaltzustand des Computers vorliegt?

Dann zeigt er auf die Adresse 65126 ($FE66), wo ein Teil der NMI-Routine zu finden ist (Siehe auch das Flußdiagramm in Bild 2):



Bild 2. Auf diese Weise verläuft ein unvorhergesehener BRK im Sande

65126 **JSR 64789** Sprung ins Programm RESTOR, in dem alle Vektoren (788-819) gemäß einer ROM-Liste auf ihre Ausgangswerte gesetzt werden.

JSR 64931 Sprung in das Programm I/O-RESET. In diesem Programm werden die beiden CIAs auf die Anfangswerte gestellt.

JSR 58648 Sprung in ein Programm, welches zuerst den VIC-II-Chip initialisiert, dann einen Bildschirmeditor-RESET durchführt. Nach Beenden dieser Routine ist der Bildschirm gelöscht.

JMP (40962)

Mit diesem indirekten Sprung ist die BRK-Unterbrechung beendet. Man sieht aber jetzt schon deutlich, daß es sich hier nicht um eine Unterbrechung im eigentlichen Sinn handelt, vielmehr um einen Abbruch. In 40962/40963 steht die Adresse des Basic-Warmstarts (58235). Danach befindet sich der Computer im READY-Zustand in der Eingabe-Warteschleife.

Das Zurückholen der Register und ein RTI erübrigt sich hier, weil ohnehin viele Werte aus dem unterbrochenen Programm inzwischen weitgehend zerstört sind und alle Unterbrechungskontrollregister (CIAs und VIC-II-Chip) neu belegt wurden. Ein unkontrollierter BRK hat also recht fatale Folgen!

Was macht ein NMI?

Wenden wir uns nun der Firmware zu, die zur Bearbeitung eines NMI vorgesehen ist (Dazu sehen Sie sich bitte in Bild 3 das Flußdiagramm an).

In der letzten Folge erfuhren wir, daß auch für diese Unterbrechung am Ende des Speichers ein Vektor vorhanden ist, nämlich $FFFA/FFFB (65530/65531). Dort steht die Adresse 65091 ($FE43), die nun in den Programmzähler gelangt und damit startet das folgende Programm:

65091 **SEI** Unterbrechungen niedrigerer Priorität werden gesperrt.

JMP (792)

Das ist nun wieder ein für uns sehr interessanter Vektor 792/793 ($318/319), der — weil er im RAM-Bereich liegt — verstellbar ist. Genau das haben wir am Ende der letzten Folge getan mittels des M-Kommandos von SMON um den NMI zu testen, den wir mit der RESTORE-Taste ausgelöst haben. Der voreingestellte Wert in diesem Vektor ist die Adresse 65095 ($FE47), also direkt der nächste Befehl nach dem indirekten Sprungbefehl.

65095 **PHA** **TXA** **PHA** Ebenso wie vorhin beim IRQ werden hier die Inhalte des Akku und der Register auf den Stapel geschoben.

NMI
IRQ sperren
Register auf Stapel
CIA-NMI sperren
NMI-CIA-Kontrollregister laden
War CIA2 der NMI-Auslöser? — J / N
Prüfung auf Modul
Modul vorhanden? — J: Modulstart / N
Tastaturabfrage vorbereiten
STOP Abfrage der Stop-Taste
Nur Restore-Taste gedrückt? — N: Programmabbruch / J
RS232C-Behandlung
Register vom Stapel holen
RTI

Bild 3. Flußdiagramm zum Ablauf einer NMI-Unterbrechung

	TYA	
	PHA	
	LDA #127	das ist binär 01111111.
	STA 56589	Sperrt alle weiteren NMI-Anforderungen
	LDY 56589	NMI-CIA Kontrollregister laden.
	BMI 65138	Wenn der NMI von der RESTORE-Taste kam, ist Bit 7 des Registers = 0, sonst = 1 (bei NMI-Anforderung durch NMI-CIA). Wenn also nicht durch die RESTORE-Taste, erfolgt Sprung.

An dieser Stelle läuft nun das Programm weiter, wenn die RESTORE-Taste der NMI-Auslöser war:

65110	**JSR 64770**	Das ist ein Unterprogramm, welches prüft, ob ein Modul ab $8000 vorhanden ist.

Dies wird dadurch angezeigt, daß von $8004 bis $8008 die Werte stehen: 195, 194, 205, 56, 48 (das ist "CBM80).

	BNE 65118	Wenn kein Modulprogramm ab $8000 vorliegt, erfolgt ein Sprung.
	JMP (32770)	Falls Modul.

Wenn ein Modul angezeigt wurde, erfolgt der indirekte Sprung nach den Vektor $8002/8003, der vom Modul vorgegeben wird. Das kann man auch nutzen, um eigene Maschinenprogramme zu starten durch einen Druck auf die RESTORE-Taste. Man muß dann nur in die Speicherstellen $8002 bis 8008 die geforderte Zieladresse beziehungsweise »CBM80« schreiben.

Der nun folgende Abschnitt wird nur angesprungen, wenn die RESTORE-Taste der NMI-Auslöser war:

65118	**JSR 63164**	Das ist ein Programmteil, der auch schon von der IRQ-Routine (nach dem Weiterstellen von TI$) durchlaufen wird. Hier werden einige Voreinstellungen für die Tastaturabfrage erledigt, die insbesonders die RUN/STOP-Taste betreffen.
	JSR 65505	Kernalroutine STOP.

Dort befindet sich ein indirekter Sprung über den Vektor 808/809 ($328/329), also auch ein verstellbarer RAM-Vektor. Im Normalfall zeigt dieser Vektor auf 63213 ($F6ED). Dort wird geprüft, ob die RUN/STOP-Taste gedrückt ist. Eine andere Methode zum Ausschalten des RUN/STOP bietet sich hier an, die die Uhr TI$ ungeschoren läßt.

	BNE 65138	Falls nur die RESTORE-Taste (also ohne RUN/STOP) gedrückt ist, erfolgt nun ein Sprung.

Waren aber sowohl die RUN/STOP- als auch die RESTORE-Taste gedrückt, dann folgt nun ein Programmabbruch, der uns schon von BRK her bekannt ist. Hier wie dort endet das Ganze dann mit dem Reset der I/O-Bausteine, des VIC-II-Chips, der Vektoren, des Bildschirmeditors und das Ergebnis ist ein Basic-Warmstart.

Ab 65138 befindet sich der Rest der NMI-Routine, auf die das Programm läuft, wenn

1) die NMI-Anforderung nicht von der RESTORE-Taste kommt oder
2) zwar von dieser Taste kommt, aber die RUN/STOP-Taste nicht gedrückt ist.

65138 bis 65211		Dieser ganze Abschnitt ist zur Behandlung der RS232C-Schnittstelle eingerichtet.
65212	**PLA**	Abschluß des NMI durch Rückschreiben des Akku und der Register vom Stapel
	TAY	
	PLA	
	TAX	
	PLA	
65217	**RTI**	Rückkehr zum unterbrochenen Programm.

Wenn Sie sich nun mal unser kleines Demo-Programm aus der letzten Folge ansehen, dann werden Sie feststellen, daß der Programmteil bis $600E lediglich den ersten Teil der normalen NMI-Routine kopiert. Die Prüfung auf das Modul und die RUN/STOP-Taste werden übersprungen. Statt dessen erfolgt nach der Abarbeitung des für die RESTORE-Taste gebauten Programmes das Ende der NMI-Routine ($FEBC = 65212). Im anderen Fall, wenn also die RESTORE-Taste nicht der Auslöser des NMI war, wird in die normale Routine ab 65138 eingemündet.

Eigentlich keine Unterbrechung: RESET

Weil wir alle Unterbrechungen hier bearbeiten wollen, soll auch der RESET angesprochen werden. Es handelt sich dabei aber nicht um eine Unterbrechung im bisher definierten Sinn. Mir fällt allerdings kein Platz ein in dieser Serie, wo der RESET besser hinpassen würde. Ähnlich wie bei NMI und IRQ wird auch hier ein Vektorinhalt in den Programmzähler geladen, der in den höchsten Speicheradressen zu finden ist (Auch hierzu wieder ein Flußdiagramm in Bild 4).

Dieser Vektor liegt in $FFFC/FFFD. Der Inhalt ist die Adresse 64738 ($FCE2) und genau dort geht das Programm dann weiter:

64738	**LDX #255**	Im ersten Teil wird der Stapelspeicher initialisiert.
	SEI	Verhindern von IRQ
	TXS	Stapelzeiger auf $FF
	CLD	Dezimal-Modus ausschalten (falls er eingeschaltet war).
	JSR 64770	Das ist wieder das Unterprogramm, das auf ein Modul prüft.

Hier ergibt sich die Möglichkeit, auch beim RESET einzugreifen, indem man die Kennung CBM80 an die abgefragten Orte packt.

	BNE 64751	Falls kein Modul, erfolgt Sprung.
64748	**JMP(32768)**	

Dieser indirekte Sprung erfolgt nach dem Vektorinhalt von $8000/8001 = 32768/32769. Das ist ein anderer Vektor als wir ihn vorhin beim NMI hatten (dort war es $8002/8003 = 32770/32771). So kann ein anderer Programmteil angesteuert werden als durch den NMI, was übrigens auch dringend erforderlich ist, weil der Stapelzeiger zerstört wurde.

64751		Hier läuft das Programm weiter, falls keine Modulkennung erkannt wurde.

Der ganze Rest dient dem Versetzen des Computers in den Einschaltzustand. Allerdings bin ich davon überzeugt, daß noch irgendein Unterschied bestehen muß zwischen dem einfachen Aus- und wieder Anschalten des Computers und einem RESET. Es hat sich nämlich bei einigen Programmen gezeigt, daß sie nach einem RESET fehlerhafte Verläufe nehmen können, was nach einem totalen Aus- und wieder Anschalten nicht zu beobachten war. Der Grund für diesen Unterschied liegt (für mich) noch im Dunkel. Vielleicht weiß das ja jemand von Ihnen. Dann schreiben Sie doch mal!

In der nächsten Folge werden wir der Sache mit dem Modulstart noch etwas weiter auf den Grund gehen und auch ein interessantes Programm dazu entwickeln. (Heimo Ponnath/gk)

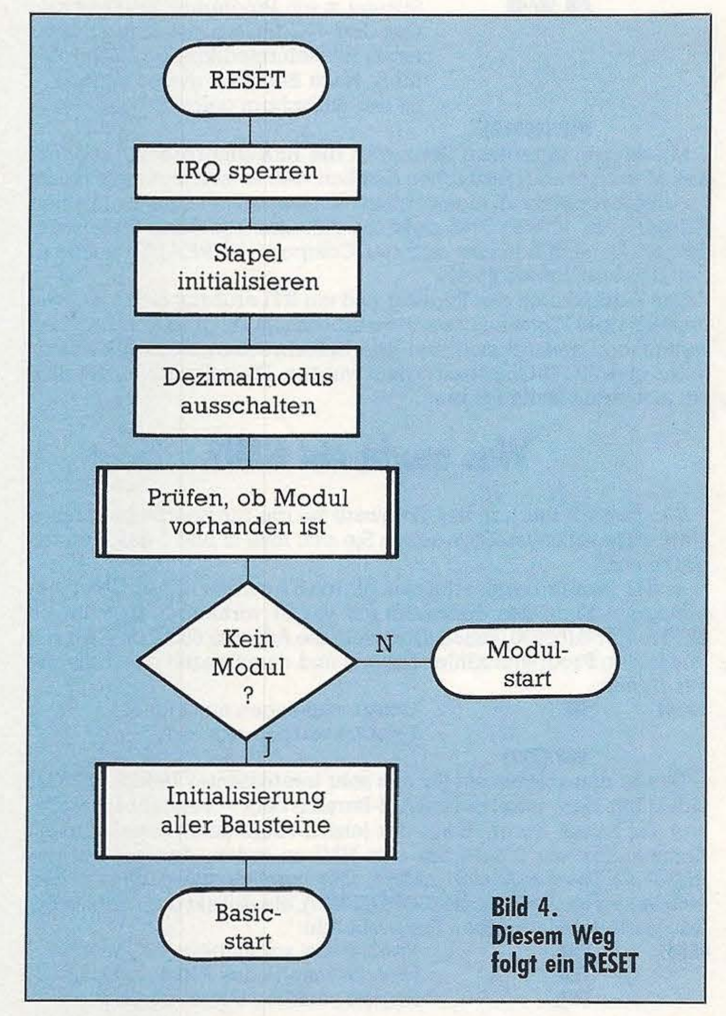


Bild 4. Diesem Weg folgt ein RESET

Memory Map mit Wandervorschlägen (9)

Die Adressen 80 bis 143, die heute behandelt werden, benutzt der Basic-Interpreter und das Betriebssystem für Stringoperationen und zur Auswertung arithmetischer Ausdrücke. Auch die interessante Charget-Routine ist dabei.

Aufmerksamen Lesern wird es nicht entgangen sein, daß wir uns bei der Wanderung durch die Speicherlandschaft des C 64 beziehungsweise VC 20 selbst überholt haben. Versehentlich wurden die Adressen 80 bis 143, die heute behandelt werden, unterschlagen. Das nächste Mal werden wir dann wieder in der richtigen Reihenfolge fortfahren.

Adresse 80 — 82 ($50 — $52)

Zeiger auf einen provisorischen Speicherplatz einer Zeichenkette, die gerade bearbeitet wird

Die Teilprogramme (von Programmierern »Routinen« genannt) des Basic-Übersetzers im ROM des Computers, welche Zeichenketten (Strings) behandeln, verwenden die ersten beiden Bytes dieser drei Speicherzellen, nämlich 80 und 81, um in Low/High-Byte-Darstellung diejenige Speicheradresse anzugeben, ab der die Zeichenkette im Programmspeicher zu finden ist.

Das dritte Byte (82) enthält die Länge der Zeichenkette. Wegen der provisorischen Natur dieses Zeigers ist er für Basic-Programme nicht geeignet.

Adresse 83 ($53)

Flagge für die Garbage Collection

In dieser Speicherzelle steht während der sogenannten Garbage Collection (Müllabfuhr) eine Zahl, die angibt, ob die Variable der zur Überprüfung anstehenden Zeichenkette eine Länge von 3 oder 7 Byte hat.

Der Vorgang der Garbage Collection ist von B. Schneider, 64'er-Ausgabe 1/85, ausführlich beschrieben worden. Angaben über die Bedeutung der Variablen einer Zeichenkette finden Sie in Teil 4 und 5 des Memory Map-Kurses (64'er-Ausgaben 2/85 und 3/85).

Adresse 84 — 86 ($54 — $56)

Sprungbefehl auf die Adressen der Basic-Funktionen

Jede Basic-Funktion, wie zum Beispiel SGN, INT, ABS, USR und so weiter, wird durch ein spezielles Teilprogramm (Routine) des Basic-Übersetzers ausgeführt. Die Anfangsadresse jeder dieser Routinen sind in einer Tabelle im ROM fest eingespeichert. Im VC 20 steht diese Tabelle von 49234 bis 49279 ($C052 bis $C07F), im C 64 von 41042 bis 41087 ($A052 bis $A07F).

In der Speicherzelle 84 steht der Sprungbefehl JMP in Maschinencode, dargestellt durch die Zahl 75 ($4C). In den beiden anderen Zellen 85/86 steht dann in Low/High-Byte-Darstellung die jeweilige Adresse in der Tabelle, welche der vom Programm gerade gebrauchten Basic-Funktion entspricht. Dieser gesamte Befehl JMP plus Adresse entspricht in Basic der GOSUB-Zeilennummer.

Ein Beispiel soll das verdeutlichen. Geben Sie direkt ein:

PRINT PEEK(84);PEEK(85);PEEK(86)

Wir erhalten
beim C 64: 76 13 184
beim VC 20: 76 13 216

Die erste Zahl ist genauso wie oben beschrieben. Die beiden anderen Zahlen ergeben zusammen die Adresse 47117 ($B80D) beziehungsweise 55309 ($D80D). Wenn Sie ein Buch mit ROM-Listing haben, werden Sie unter dieser Adresse die Routine für die Funktion »PEEK« finden. Das ist natürlich nicht erstaunlich, haben wir doch gerade vorher als letzten Befehl genau diese Funktion eingegeben.

Leider ist das auch die einzige Funktion, die ich Ihnen vorführen kann, denn zum Vorführen muß ich eben immer PEEKen, so daß beim besten Willen immer nur die oben angegebenen Zahlen erscheinen können.

Adresse 87 — 96 ($57 — $60)

Arbeitsspeicher für diverse Arithmetik-Routinen des Basic-Übersetzers

Diese zehn Speicherplätze werden von verschiedenen Teilprogrammen (Routinen), besonders bei arithmetischen Operationen, als Zwischenspeicher verschiedener Werte, Flaggen und Zeiger benützt.

Adresse 97 — 102 ($61 — $66)

Gleitkomma-Akkumulator Nr. 1

»Akkumulator« heißt seit der Zeit der mechanischen Rechenmaschinen eine Speicherzelle, welche bei Rechenoperationen dadurch im Mittelpunkt steht, daß laufend Daten in sie hineingeschrieben beziehungsweise aus ihr herausgelesen werden.

Normalerweise trägt diesen Namen das zentrale Rechenregister des Mikroprozessors. Leser des Assembler-Kurses kennen diesen Akkumulator inzwischen zur Genüge.

Die Speicherzellen 97 bis 102 werden deswegen ebenfalls Akkumulator genannt, weil sie bei der Verarbeitung von Gleitkommazahlen eine ähnliche zentrale Rolle spielen.

Am Anfang dieses Kurses in Ausgabe 11/84, bei der Beschreibung der Adressen 0 bis 2 des VC 20 (784 bis 786 beim C 64), und dann noch einmal im 64'er, Ausgabe 12/84, bei den Adressen 3/4 und 5/6 habe ich Ihnen versprochen, im Detail auf die Darstellung von Gleitkommazahlen und auf die Verwendung des damals schon genannten Gleitkomma-Akkumulators einzugehen. Heute wäre es nun soweit.

Inzwischen hat Herr Ponnath im Assemblerkurs mir die Arbeit aber bereits abgenommen. Im Teil 8 des Kurses (64'er, Ausgabe 4/85) finden Sie alle Details zu diesem Thema.

Hier soll uns eine kurze, zusammenfassende Bemerkung genügen.

Zelle 97 enthält den Exponenten. Die Zellen 98 bis 101 enthalten die Mantisse.

Zelle 102 enthält das Vorzeichen der Gleitkommazahl. Eine 0 bedeutet ein positives, die Zahl 255 ein negatives Vorzeichen.

Mit dem Gleitkomma-Akkumulator Nr. 1 sind zwei weitere Speicherzellen eng verbunden, nämlich 104 ($68) und 112 ($70).

Ganz zum Schluß ist noch erwähnenswert, daß nach der Umwandlung einer Gleitkommazahl in eine ganze Zahl diese als Low/High-Byte in den beiden Speicherzellen 98 und 99 steht, was für Maschinenprogramme vielleicht recht nützlich sein kann.

Adresse 103 ($67)

Zwischenspeicher beziehungsweise Zählregister

Diese Adresse wird von zwei Routinen verwendet. Der Basic-Übersetzer benützt sie als Vorzeichenspeicher bei der Umwandlung von Zahlen aus dem ASCII-Format in Gleitkommazahlen. Das Betriebssystem verwendet diese Adresse als Zähler der Abarbeitungsschritte bei der Berechnung eines Polynoms der Form

y = a0 + a1*x + a2*x↑2 + a3*x↑3 + ...

Adresse 104 ($68)

Überlauf-Speicher des Gleitkomma-Akkumulators Nr. 1

Wenn eine Zahl so groß wird, daß sie mit den zur Verfügung stehenden Stellen nicht mehr dargestellt werden kann, sprechen wir von einem »Überlauf«.

Bei Gleitkommazahlen liegt diese Überlaufgrenze bei $1,70141183 * 10^{38}$.

Während einer mathematischen Berechnung kann es intern im Computer vorkommen, daß ein Überlauf eintritt, der aber am Ende der Operation wieder verschwinden würde. Der Akkumulator Nr. 1 benützt in einem derartigen Fall die Speicherzelle 104, um die verfügbare Stellenzahl um 8 Bit zu vergrößern. Für endgültige Resultate steht diese Erweiterung natürlich nicht zur Verfügung.

Dieser Vorgang tritt besonders häufig bei der Umwandlung von ganzen Zahlen oder Zeichenketten in Gleitkommazahlen auf.

Adresse 105 — 110 ($69 — $6E)

Gleitkomma-Akkumulator Nr. 2

Spätestens jetzt verstehen Sie, warum der Akkumulator der Speicherzellen 97 bis 102 die Nr. 1 hat. Es gibt hier noch einen zweiten Gleitkomma-Akkumulator, der ein identischer Zwilling ist. Zwei Akkumulatoren sind immer dann notwendig, wenn mathematische Operationen ablaufen, welche mehr als einen Operanden verarbeiten, wie zum Beispiel Multiplikation, Division und so weiter.

Aufgrund der Identität der beiden Akkumulatoren kann ich mir eine weitere Beschreibung ersparen.

Adresse 111 ($6F)

Flagge für Vorzeichenvergleich der Gleitkomma-Akkumulatoren Nr. 1 und Nr. 2

Wenn die Zahl in beiden Akkumulatoren gleiche Vorzeichen haben, steht in Speicherzelle 111 eine 0, bei verschiedenen Vorzeichen eine 255.

Adresse 112 ($70)

Rundungsspeicher des Gleitkomma-Akkumulators Nr. 1

Es kann vorkommen, daß die Mantisse einer Gleitkommazahl mehr Stellen hat, als mit den vier Mantissen-Bytes des Akkumulators Nr. 1 (Zellen 90 bis 101) dargestellt werden können. In diesem Fall werden die hintersten, das heißt die unwichtigsten Stellen hinter dem Komma in der Zelle 112 abgelegt. Von dort werden sie geholt, um die Genauigkeit von mathematischen Operationen zu erhöhen und auch, um Endresultate abrunden zu können.

Adresse 113 — 114 ($71 — $72)

Zwischenspeicher für verschiedene Routinen

Diese Speicherzellen werden von sehr vielen Routinen des Übersetzers und des Betriebssytems, wie zum Beispiel Zeichenkettenverarbeitung, interne Uhr (TI$), Bestimmung der Größe von Feldern (Arrays) und etlichen anderen verwendet.

Adresse 115 — 138 ($73 — $8A)

Teilprogramm »Nächstes Zeichen eines Basic-Textes holen« (CHRGET-Routine)

Die Problematik der Übersetzung von Basic-Befehlen und Anweisungen besteht darin, daß die Übersetzungsschritte durch entsprechende Programmteile des Basic-Übersetzers im Computer fest vorprogrammiert sein müssen, was bedeutet, daß diese Programme natürlich im — nicht veränderbaren — ROM stehen.

Auf der anderen Seite verlangt aber der Übersetzungsvorgang, daß gewisse Teile dieser Programme sich laufend verändern. Als Beispiel soll der Zeiger herhalten, der angibt, in welcher Speicherzelle das nächste zu bearbeitende Zeichen steht. Dieser Zeiger und die zusammengehörigen Programmschritte dürfen natürlich nicht im ROM stehen, denn da sind sie ja nicht änderbar.

Dieser Konflikt wird dadurch gelöst, daß dieses »variable« Teilprogramm des Übersetzers zwar im ROM steht (im C 64 ab 58274 oder $E3A2, im VC 20 ab 58247 oder $E387), von wo es aber direkt nach dem Einschalten des Computers in das RAM, und zwar in die Speicherzellen 115 bis 138, umgeladen wird.

Dieses Teilprogramm, welches die Zeichen zur Übersetzung herbeiholt und deswegen »Character-Get« oder kurz CHARGET-Routine genannt wird, ist wegen seiner Veränderbarkeit natürlich ein beliebtes Objekt aller möglichen Manipulationen. Es ist deshalb im Assembler-Kurs, Teil 5, im 64'er, Ausgabe 1/85, im Detail beschrieben worden, allerdings mit Schwerpunkt auf Assembler/Maschinensprache.

Für Basic-Programmierer möchte ich hier deshalb eine kurze Beschreibung der CHRGET-Routine einfügen.

Die Routine beginnt mit einem Sprung auf den oben schon erwähnten Zeiger in Adresse 122/123, welcher seinerseits auf die Adresse zeigt, in welcher das nächste zu übersetzende Zeichen steht. Das Zeichen wird entsprechend dem Hinweis des Zeigers geholt, in den Akkumulator des Mikroprozessors geladen und dort verschiedenen Prüfungen unterzogen. Ist das Zeichen ein Gänsefuß, erkennt das Programm, wie es das nächste Zeichen interpretieren und behandeln muß. Ein Doppelpunkt leitet einen neuen Befehl ein, eine Leerstelle wird unterdrückt und so weiter.

Mit dem Befehl

```
PRINT PEEK(122) + 256*
PEEK(123)
```

können wir innerhalb eines Programms ausdrucken, wohin der Zeiger nach dem letzten Basic-Zeichen deutet. Eine Überprüfung mit den Methoden, die ich bei der Besprechung der Speicherzellen 43 bis 56 genannt habe, zeigt Ihnen den Zusammenhang.

Normalerweise wird der Zeiger in 122/123 nach jedem Zeichen um 1 erhöht, da ja die Zeichen einer Basic-Zeile hintereinander im Speicher stehen. Ein GOTO- oder GOSUB-Befehl kann diese Folge natürlich unterbrechen, ebenso wie eine willkürliche Änderung durch einen Eingriff von außen.

Ein derartiger Eingriff, auch »wedge« (Keil) genannt, öffnet natürlich Tür und Tor für Programmiertricks, insbesondere für Einbau von neuen, selbsterfundenen Befehlen. Man kann entweder den allerersten Sprungbefehl auf den Zeiger so umlenken, daß er auf ein eigenes Maschinenprogramm springt, oder man kann den Zeiger selbst »verbiegen«, so daß er auf eine andere Adresse und damit auf ein anderes Zeichen zeigt. Es gibt dafür viele Möglichkeiten, die aber alle nur in Maschinencode funktionieren. Theoretisch können wir natürlich den Inhalt des Zeigers in 122/123 durch POKE verändern. Aber was dann? Jeder nachfolgende Basic-Befehl löst natürlich wieder die normale Übersetzungsroutine aus und unser schöner POKE ist für die Katz.

Wie so ein Wedge in Maschinensprache gemacht wird, hat Christoph Sauer im VC 20-Kurs — 64'er, Ausgabe 9/84 — beschrieben. Allerdings ist das Beispiel für Anfänger nicht verständlich, was mich zu der Überzeugung bringt, daß die CHRGET-Routine und ihre Anwendung einen eigenen Aufsatz wert wäre.

Adresse 139 — 143 ($8B — $8F)

Wert der RND-Funktion als Gleitkommazahl

Mit dem Befehl RND(X) kann bekanntlich eine Zufallszahl erzeugt werden. Was das bedeutet und wie »zufällig« diese Zahlen sind, können Sie dem nebenstehenden Texteinschub »Wie zufällig sind Zufallszahlen« entnehmen.

Beim Einschalten des Computers werden die Zahlen 128, 79, 199, 82 und 88 in diese Speicherzellen geschrieben. Mit der folgenden Zeile können Sie das gleich nach dem Einschalten des Computers leicht überprüfen.

```
FOR X=139 TO 143:PRINT
PEEK(X):NEXT
```

Nach den Manipulationen des RND-Befehls wird das Resultat wieder in die Zellen 139 bis 143 als neuer Ausgangswert (seed) für den nächsten RND-Befehl gebracht.

Diese fünf Zahlen stellen eine Gleitkommazahl dar. Ihre Form entspricht dabei der Aufteilung, wie sie auch im Gleitkomma-Akkumulator (97 bis 101) verwendet wird.

Eine Abfrage dieser Zahlen aus den Zellen 139 bis 143 ist natürlich möglich, aber nicht ergiebig, weil das Resultat von RND(X) direkt als Zahl verfügbar ist, während die 5 Bytes erst ein eine brauchbare Zahl umgerechnet werden müßten. Eine Änderung durch POKEn neuer Werte in diese Speicherzellen geht leider nicht.

Adresse 144 ($90)

Ein-/Ausgabe-Status ST

Ab dieser Speicherzelle wurde der Kurs irrtümlicherweise, wie schon erwähnt, im 64'er, Ausgabe 6/85, weitergeführt. Hiermit haben wir die Lücke geschlossen und den Anschluß erreicht.

Das nächste Mal werden wir dann wieder in der richtigen Reihenfolge fortfahren.

(Dr. H. Hauck/ah)

Wie zufällig sind Zufallszahlen?

Der Befehl RND(X) ergibt eine Zufallszahl zwischen 0 und 1 — so steht es im Commodore-Handbuch.

Eine Zufallszahl ist definitionsgemäß rein dem Zufall überlassen, ihr Wert kann nicht vorhergesehen werden. Wie kann aber ein Computer, in dem alle Vorgehensweisen und Arbeitsschritte fest vorprogrammiert sind, eine zufällige Zahl erzeugen? Die Commodore-Computer machen das so:

Der Befehl RND nimmt eine bestimmte Ausgangszahl (auf die ich noch näher eingehen werde), auf englisch »seed« = Samen genannt, multipliziert sie mit 11879546.4 und zählt $3.92767778 * 10^8$ dazu. Die 5 Bytes der resultierenden Gleitkommazahl werden miteinander vertauscht und in einen positiven Bruch umgewandelt. Diese Manipulation ergibt die »Zufallszahl«, die als neuer »Samen« in den Speicherzellen 139 — 143 gespeichert wird.

Es ist sicher einzusehen, daß die Zufälligkeit nicht sehr hoch sein kann, es sei denn, die oben genannte und noch nicht erklärte Ausgangszahl ist zufällig.

Die erste Ausgangszahl hängt vom »Argument« des RND(X)-Befehls ab, das heißt vom Wert X, der in der Klammer dahinter steht. Es gibt drei Möglichkeiten für das Argument:

— eine positive Zahl (egal, welcher Wert)
— eine negative Zahl
— die Zahl 0

Eine positive Zahl

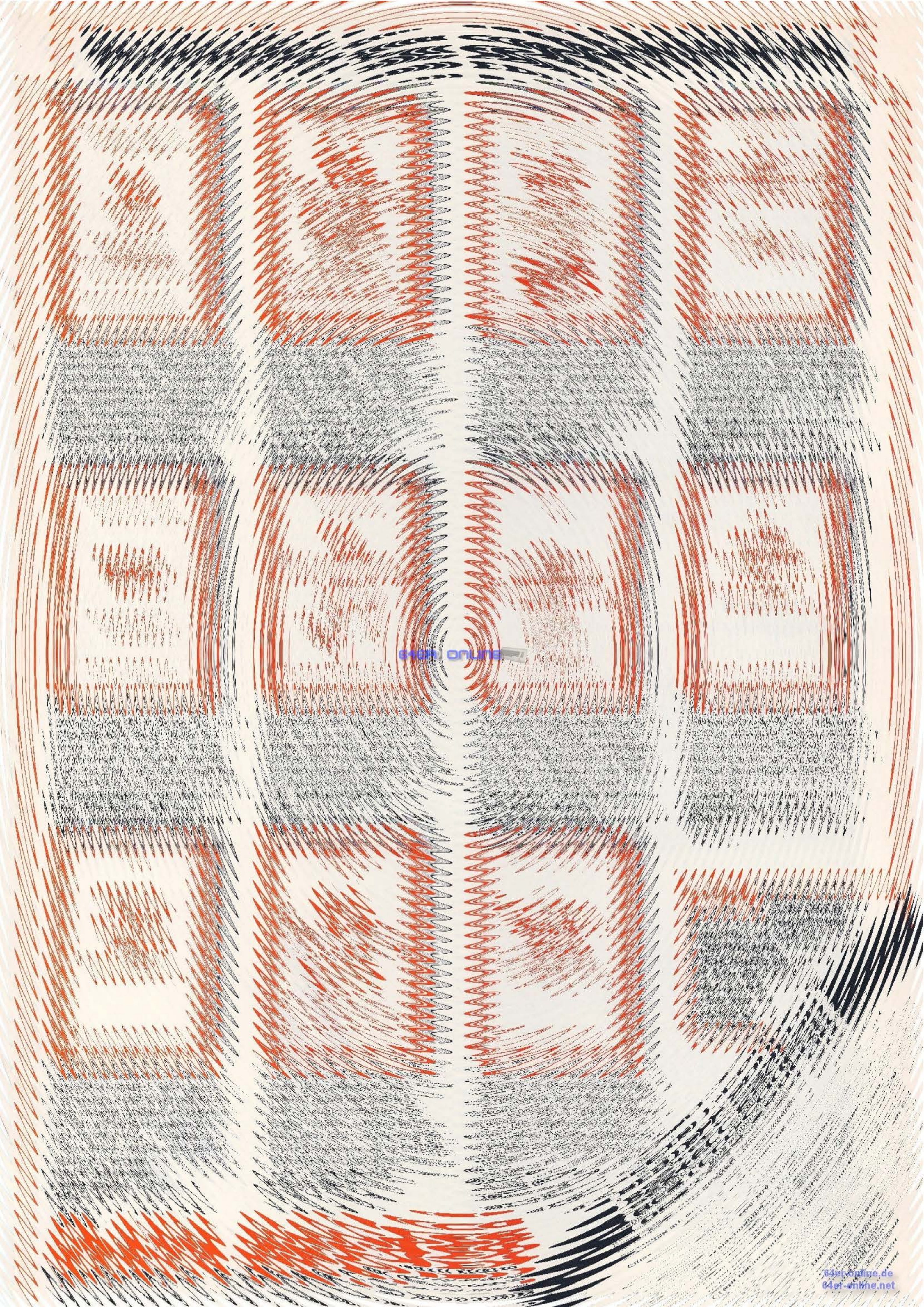
zum Beispiel RND (1) oder RND(56) nimmt als Samen die Zahl 0.811635157, die beim Einschalten des Computers als 5-Byte-Gleitkommazahl in die Speicherzellen 139 bis 143 geschrieben worden ist. In den fünf Zellen stehen die Zahlen 128, 79, 199, 82, 88.

Daraus folgt aber, daß nach dem Einschalten des Computers mit RND(1) immer dieselbe Sequenz von Zufallszahlen erzeugt wird. Schalten Sie bitte den Computer aus und ein und geben Sie ein:

```
10 PRINT RND(1):GOTO 10
```

Notieren Sie die ersten paar Zahlen und wiederholen Sie mit Aus-/Einschalten die Prozedur. Sie werden immer dieselben Zahlen sehen.

Zum Austesten von Programmen mit bekannten Zahlensequenzen ist diese Methode sicher wichtig, aber echte Zufallszahlen sind das nicht!



Eine negative Zahl
zum Beispiel RND(—1) oder RND(—95) bringt als erstes das Argument selbst (in meinem Beispiel —1 oder —95) als Gleitkommazahl in die Speicherzellen 139 bis 143, von wo sie als Samen den schon beschriebenen Manipulationen unterworfen wird. Nur — mit einem bestimmten negativen Argument erhalten Sie immer dieselbe Zufallszahl. Probieren Sie es aus:
PRINT RND(—2) ergibt immer dieselbe Zahl.

Es mag Fälle geben, wo die Vorgabe des allerersten Samens wünschenswert ist. Ich will aber von zufälligen Zahlen sprechen. Wir können diese Methode des negativen Arguments dadurch verbessern, daß wir als Argument selbst eine Zufallszahl nehmen.

Als derartige Zahl bietet sich der Wert der inneren Uhr TI an, die beim Einschalten des Computers losläuft und 60mal in der Sekunde weitergestellt wird. Da kein Mensch wissen kann, welchen Wert die Uhr TI gerade hat, kommt der Befehl RND(—TI) dem absoluten Zufall schon sehr nahe.

Das Argument (0)
verwendet eine andere Methode. Als Samen nimmt er eine sich ständig ändernde Zahl, die beim VC 20 aus vier Registerinhalten des VIC-Interface-Bausteins genommen werden. Beim C 64 wird es ähnlich gemacht, nur ist der VIC-Baustein ein anderer Typ.

Mit derselben Methode nach dem Einschalten wie im ersten Fall oben, können Sie das leicht überprüfen.

Ich habe eingangs zitiert, daß RND(X) eine Zahl zwischen 0 und 1 erzeugt; das gilt aber nur für ein positives Argument. Wenn Sie hingegen eine Zufallszahl innerhalb eines ganz bestimmten Bereiches brauchen, müssen Sie anders vorgehen.

Kochrezept Nr. 1

Mit folgender Formel ist der Zahlenbereich beliebig vorgebbar:
X=(RND(1)*A)+B

Die Zahl A gibt einen Bereich von 0 bis A vor.

Die Zahl B legt den untersten Wert des Bereiches fest.

Beispiele:
10 PRINT (RND(1)*6)+1:GOTO 10 erzeugt Zahlen von 1 bis 6
10 PRINT (RND(1)*52)+1:GOTO 10 erzeugt Zahlen von 1 bis 52
10 PRINT (RND(1)*6)+10:GOTO 10 erzeugt Zahlen von 10 bis 16

Mit dem Vorschalten der Funktion INT vor den Befehl RND werden die Zufallszahlen auf ganze Zahlen beschränkt.
10 PRINT INT (RND1)*6)+10:GOTO 10

Noch einmal: Zufallszahlen innerhalb bestimmter Zahlenbereiche sind gekoppelt mit einem positiven Argument. Wir haben aber gesehen, daß gerade so keine echten Zufallszahlen erzeugt werden. Deshalb brauchen wir noch ein zweites Kochrezept.

Kochrezept Nr. 2

Wenn Sie in einem Programm nach dem Einschalten des Computers immer neue Zufallszahlen brauchen, ist es empfehlenswert, für die allererste Zufallszahl RND(—TI) oder RND(0) zu verwenden, dann aber mit RND(1) fortzufahren.

Dasselbe gilt, wenn ein Programm wegen INPUT oder WAIT eine Pause hat. Nach der Pause sollte zuerst ein neuer Ausgangswert genommen werden.

Als letztes will ich noch beschreiben, wie man Zufallszahlen innerhalb von Maschinenprogrammen erzeugen kann.

Im Betriebssystem steht natürlich eine Routine für den Befehl RND. Im C 64 beginnt sie ab 57495 ($E097), im VC 20 ab 57492 ($E094).

Der Ausgangswert (Samen) wird dabei aus dem Gleitkomma-Akkumulator Nr. 1 geholt, dessen Vorzeichen oder Wert 0 das weitere Vorgehen der Routine bestimmt.

Sie müssen also den Samen in den Akkumulator Nr. 1 laden und dann mit JSR auf die RND-Routine springen. Als Resultat können Sie einen oder mehrere Werte der Zellen 140 bis 143 verwenden und nach Belieben weiterverarbeiten.

Fortsetzung von Seite 121

mit denen allgemein Schleifen wesentlich flexibler und übersichtlicher aufgebaut werden können. Dabei können 64 solcher Schleifen ineinander verschachtelt werden.

Weiterhin können Fehler im Programm mit dem Befehl TRAP abgefangen und mit den dazu vorhandenen Variablen EL, ER und ERR$ lokalisiert werden. Anschließend kann mit RESUME zu der Programmstelle zurückgesprungen werden, an der der Fehler auftrat.

Einige Kommandos zur Stringverarbeitung wurden ebenfalls eingebaut, wie etwa INSTR zur Stringsuche. Interessant ist hier die Erweiterung des Befehls MID$, der jetzt bei Zuweisungen auch links vom Gleichheitszeichen stehen kann und somit ein kontrolliertes Einsetzen von Zeichenketten in andere ermöglicht.

Die gesamte Speicherverwaltung wurde, wie am Anfang erwähnt, umgekrempelt. Deshalb wurden Befehle wie POKE und PEEK mit geändert und greifen jetzt nur noch auf den 64-KByte-RAM-Speicher zu. Um trotzdem alle Ein-/Ausgabebausteine zu erreichen, wurden einige fest installierte Variablenfelder eingerichtet. Man kann zum Beispiel mit den Variablen VIC(x) auf alle Register des Grafikbausteins direkt zugreifen, ohne einen einzigen POKE-Befehl verwenden zu müssen. Ähnliches gilt für den Soundchip und die beiden CIAs. Ferner läßt sich der Bildschirm- und Farbspeicher mit den Variablen VID(x) und COL(x) beeinflussen, wobei x von 0 bis 999 reicht. Daneben gibt es die Variablen BORDER, PAPER und INK, mit denen man die Bildschirmfarben direkt beeinflussen kann. Der Nachteil dieses Konzeptes ist es, daß Programme, die in Basic V2.0 geschrieben wurden und viele POKEs verwenden, oft nicht funktionieren.

Abgerundet wird die Palette der neuen Befehle durch Kommandos wie zum Beispiel UPPER und LOWER zum Festlegen des Groß- oder Kleinschriftmodus, CLS zum Löschen des Bildschirmes, RESET um den Einschaltzustand zu erreichen und MONITOR zum direkten Aufruf eines vorher eingeladenen Monitors, wenn dieser über einen BREAK-Einsprungspunkt verfügt. Auch der OLD-Befehl ist vorhanden, mit dem man ein NEW oder das oben erwähnte RESET wieder aufheben und damit ein Programm im Speicher wieder restaurieren kann. Ferner läßt sich mit dem Befehl STOP ON/OFF die Stop-Taste blockieren oder freigeben.

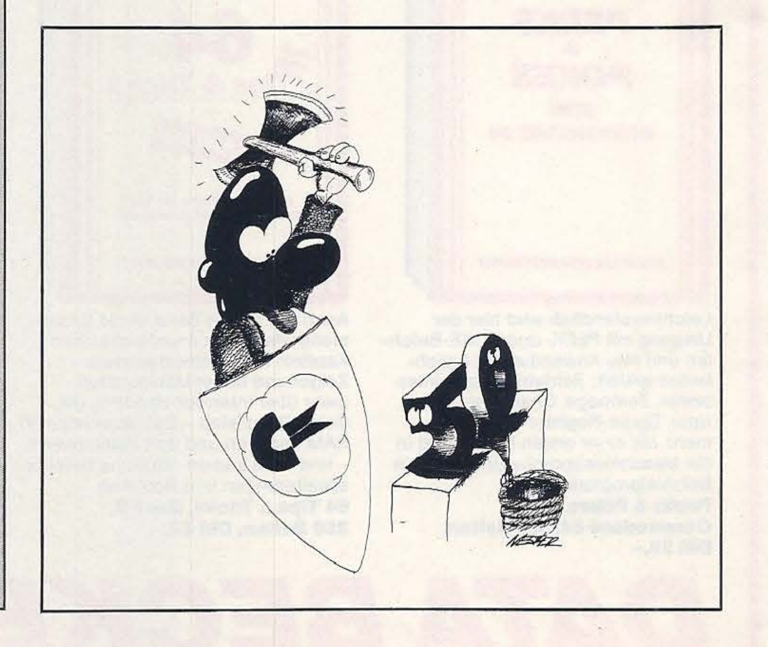
Das Handbuch erläutert all diese Befehle in verständlicher Art mit vielen Programmbeispielen.

Bemerkenswert ist in diesem Zusammenhang, daß Erweiterungen wie TurboAccess oder Hypra-Load (ROM-Version) weiterhin funktionieren.

Business Basic ist sowohl wegen der Fähigkeit mehr als 61 KByte direkt in Basic zur Verfügung zu stellen, als auch wegen seiner Qualitäten als Basic-Erweiterung sehr interessant. Nur einige Befehle zur Steuerung der hochauflösenden Grafik, die ja durchaus im Sinne dieser Erweiterung liegen würden, fehlen etwas. Der Name dieser Erweiterung ist trotzdem durchaus berechtigt, denn Business Basic erlaubt professionelles Programmieren zum Heimanwender-Preis (198 Mark).

(K. Hinsch/A. Wängler/rg)

Info: Kingsoft, Schnakebusch 4, 5106 Roetgen, Tel. 02408/8319, Preis: 198 Mark



Dem Klang auf der Spur (7)

Der vorliegende Abschnitt wird dem Titel dieser Reihe in besonderer Weise gerecht. Anhand einiger Beispiele wird gezeigt, was man alles an Klängen und Tönen aus dem SID herausholen kann, wenn man über die Programme Modulator (Ausgabe 4 und 5/85) und Sound-Editor (7/85) verfügt.

Zunächst einige allgemeine Tips zur Suche nach Klängen und Effekten.

Der Sound-Editor eignet sich eigentlich gut zum Experimentieren durch Versuch und Irrtum, da das Resultat immer unmittelbar hörbar wird. Dennoch sollte man sich über die Wirkung der vielen möglichen Einstellungen schon vorher im klaren sein, um gezielt experimentieren zu können. Dazu einige praktische Hinweise unter Verzicht auf theoretischen Ballast:

1. Mit der Grundeinstellung anfangen

Wenn man eine ganz neue Einstellung machen möchte, beginnt man am besten mit der Grundeinstellung, die nach dem Start des Programmes vorliegt. Es ist nur Stimme 1 mit einem völlig unmodulierten, symmetrischen Rechteckklang hörbar. Die voreingestellte Hüllkurve hat einen leicht perkussiven Charakter (kurzes Attack/Anschwellen) und mittellanges Decay (Halten) und eine angenehme Ausklingphase (mittellanges Release).

2. Kurvenform und Hüllkurve wählen

Man beginnt mit dem Untermenü Stimme (Shift V). Dort kann man die wichtigsten SID-Parameter, wie Kurvenform, Hüllkurve und Tonlage, einstellen. Die Pulsweite PW ist nur bei der Rechteckkurve relevant. Der voreingestellte Wert 2048 entspricht einem symmetrischen Rechteck. Es ist empfehlenswert, von dieser Einstellung abzuweichen, da unsymmetrische Rechtecke obertonreicher sind und damit meistens interessanter klingen. Der Portamento-Effekt ist übrigens nur bei Werten größer als 0 aktiv.

3. Kurvenformen kombinieren

Interessante und sehr obertonreiche Klänge erhält man, wenn man im CONTROL-Register das Bit für Rechteck zusammen mit dem für Dreieck oder Sägezahn setzt. Man muß dabei auch etwas mit der Pulsweite experimentieren, da bei bestimmten PW-Werten überhaupt nichts hörbar ist. Versucht man, Rauschen mit einer anderen Kurvenform zu kombinieren, kann es passieren, daß sich der Oszillator »aufhängt«. In diesem Fall muß das TEST-Bit kurz ein- und ausgeschaltet werden, damit wieder etwas hörbar wird.

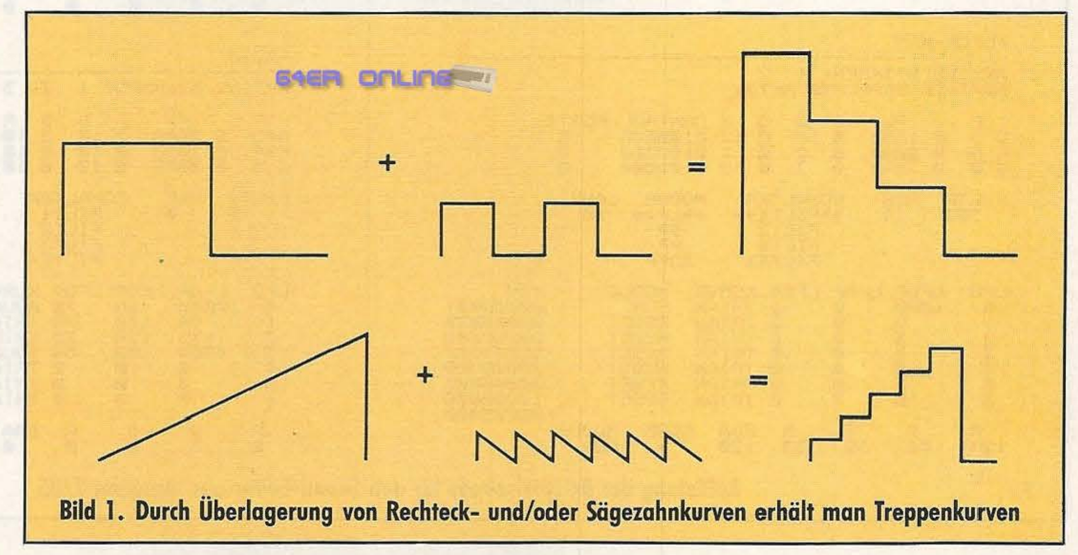


Bild 1. Durch Überlagerung von Rechteck- und/oder Sägezahnkurven erhält man Treppenkurven

4. Sustain-Modus

Oft ist es erwünscht, daß ein Ton oder Klang auch ohne Tastenbetätigung weiterklingt, zum Beispiel wenn man in Ruhe die Wirkung von Parameteränderungen studieren möchte. Dazu ist der Sustain-Modus vorgesehen. Man muß dazu nur die Taste »S« betätigen. Eine weitere Betätigung schaltet diesen Modus wieder aus.

5. Filter

Das Filter ist eine etwas problematische Komponente des SID. Versuche mit mehreren C 64-Exemplaren haben ergeben, daß Klangeinstellungen, die das Filter einbeziehen, von Gerät zu Gerät unterschiedlich klingen. Bei gleichen Einstellungen machen die verschiedenen SID-Filter unterschiedlich auf oder zu. Es kann daher vorkommen, daß die Sounds mit den hier abgedruckten Parametersätzen nicht »gut klingen«, sofern sie das Filter verwenden. Abhilfe schafft man dadurch, indem man die Filterfrequenz nach oben oder unten korrigiert, bis das Resultat zufriedenstellend klingt. Eine Filterwirkung ist nur dann hörbar, wenn mindestens eine Stimme auf das Filter geschaltet wird (Spalte SCHALTER) und wenn mindestens ein Modus (LP, BP oder HP) aktiviert ist.

6. Modulationen

Richtig interessant werden Klänge erst durch Modulationen. Damit eine Modulation wirksam wird, müssen in zwei Untermenüs Einstellungen gemacht werden. Zuerst legt man im Untermenü KSV (Kreuzschienen-Verteiler) fest, welche Modulationsquelle (LFO 0-6 oder der Soft-Hüllkurvengenerator) auf welches Ziel (Frequenzen, Pulsweiten, Filterfrequenz, Lautstärke) wirken soll. Am besten beginnt man einmal mit einer Frequenzmodulation durch einen LFO. Eine 1 in der rechten oberen Ecke der KSV-Matrix zum Beispiel schaltet LFO 0 auf die Frequenz von Stimme 1.

Dann muß die Modulationsquelle selbst eingestellt werden.

Ein LFO ist erst dann wirksam, wenn:
— die Frequenz LFOF größer als 0 ist,
— die Amplitude LFOA größer als 0 ist,
— MODUS auf RUN geschaltet ist,
— bei KURVE=SQUARE die Pulsweite LFOP größer als 0 ist.

Der Software-Hüllkurvengenerator ist wirksam, wenn:
— Attack A größer als 0 ist,
— die Amplitude EGA größer als 0 ist,
— MODUS auf RUN geschaltet ist.

Für vernünftige Hüllkurven sollten auch die Parameter Decay D und Release R größer als 0 gewählt werden. Der Soft-Hüllkurvengenerator wird durch die SPACE-Taste ausgelöst. Durch Shift-SPACE kann man die Triggerung des Soft-EG (EG = Envelope Generator) an das normale Spielen von Tönen koppeln. Ein weiteres Shift-SPACE hebt diese Kopplung wieder auf.

7. Die Tiefe der Modulationen

Die Modulationstiefe (das ist der Grad der Modulationswirkung) wird über die Amplitude der Modulationsquelle (LFOA beziehungsweise EGA) eingestellt. Für die Modulation von Pulsweite, Filterfrequenz und Lautstärke werden relativ große Modulationsamplituden benötigt, damit ein Effekt hörbar wird (etwa ab 20 aufwärts). Die Frequenzmodulation reagiert dagegen empfindlicher. Amplituden bis maximal etwa 10 werden als Vibrato empfunden, das heißt der modulierte Ton wirkt noch einheitlich in seiner Tonhöhe, wird aber breiter und lebendiger empfunden. Amplituden über 10 bewirken einen zunehmenden Heulton- oder Sirenen-Effekt. Natürlich hat auch die Modulationsgeschwindigkeit LFOF einen starken Einfluß auf das Klangbild des modulierten Tones.

8. Sprünge zwischen zwei Tönen

Durch Frequenzmodulation mit einer Rechteckkurve erreicht man, daß die Frequenz nicht auf- und abgleitet, sondern zwischen zwei Werten hin und her springt. Durch geeignete Wahl der Amplitude erhält man musikalisch sinnvolle Intervalle. Beispiele:

LFOA	Intervall
46	kleine Terz
61	große Terz
76	Quarte
104	Quinte
176	Oktave

9. Abspeichern

Man sollte nicht versäumen, die vielleicht mühevoll gefundenen Sounds abzuspeichern. Im Untermenü »Sounds« kann man bis zu 24 Sounds ablegen und mit



```
MOLL WEIT
----------------------------------------
AKTIVE STIMMEN: 1  2  3

  F     PW    A  D  S  R CONTROL PORTA
C 3  0 2048  0 13  8 13 0010000     0
D#4  0 2048  0 12  8 12 0010000     0
G 5  0 2048  0 11  8 11 0010000     0

FILTF  RES   SCHALTER   MODUS  LAUT
   0     0     FILT1      LP    12
               FILT2      BP
               FILT3      HP
               FILTEX    3OFF

LFO  LFOF LFOP LFOA KURVE  MODUS      KSV
 0   5000    0    1 TRIAN  RUN        00000001
 1      0    0    0 TRIAN  RESET      00000001
 2      0    0    0 TRIAN  RESET      00000001
 3      0    0    0 TRIAN  RESET      00000000
 4      0    0    0 TRIAN  RESET      00000000
 5      0    0    0 TRIAN  RESET      00000000
 6      0    0    0 TRIAN  RESET      00000000
                                      00000000
  A    D    S    R  EGA  FORM  MODUS
  0    0    0    0    0    +   RESET
```

```
STRINGS
----------------------------------------
AKTIVE STIMMEN: 1  2  3

  F     PW    A  D  S  R CONTROL PORTA
G 3  0 1040  5 10  8 12 0100000    80
C 2  0 1040 13 10  8 12 0100000    80
C 5 10 2048 11 10  8 10 0100000    80

FILTF  RES   SCHALTER   MODUS  LAUT
 500    15     FILT1    **LP**  15
            **FILT2**     BP
               FILT3      HP
               FILTEX    3OFF

LFO  LFOF LFOP LFOA KURVE  MODUS      KSV
 0   6800    0    8 TRIAN  RUN        00000001
 1   8000    0    8 TRIAN  RUN        00000010
 2   2600    0   90 TRIAN  RUN        00000011
 3    400    0   90 TRIAN  RUN        00000100
 4      0    0    0 TRIAN  RESET      00001000
 5      0    0    0 TRIAN  RESET      00000100
 6      0    0    0 TRIAN  RESET      00000000
                                      00000000
  A    D    S    R  EGA  FORM  MODUS
  0    0    0    0    0    +   RESET
```

```
KLICK-WAH
----------------------------------------
AKTIVE STIMMEN: 1  2
SOFT-EG-KOPPLUNG AKTIV

  F     PW    A  D  S  R CONTROL PORTA
C 3  0 1050  0 10  8 12 0100000     0
C 5  0 2048  0  2  0  1 0100000     0
G 5  0 2048  0  3  0  3 0100000     0

FILTF  RES   SCHALTER   MODUS  LAUT
 350    14  **FILT1**   **LP**  15
               FILT2      BP
               FILT3      HP
               FILTEX    3OFF

LFO  LFOF LFOP LFOA KURVE  MODUS      KSV
 0   6000    0    6 TRIAN  RUN        00000001
 1      0    0    0 TRIAN  RESET      00000000
 2      0    0    0 TRIAN  RESET      00000000
 3      0    0    0 TRIAN  RESET      00000000
 4      0    0    0 TRIAN  RESET      00000000
 5      0    0    0 TRIAN  RESET      00000000
 6      0    0    0 TRIAN  RESET      10000000
                                      00000000
  A    D    S    R  EGA  FORM  MODUS
125   50   60   30  120    +   RUN
```

```
SIRENE
----------------------------------------
AKTIVE STIMMEN: 1  2  3

  F     PW    A  D  S  R CONTROL PORTA
C 3  0 2048  0 10  8 10 0010000     1
D#3  0 2048  0 10  8 10 0010000     1
F 3  0 2048  0 10  8 10 0010000     1

FILTF  RES   SCHALTER   MODUS  LAUT
   0     0     FILT1      LP    15
               FILT2      BP
               FILT3      HP
               FILTEX    3OFF

LFO  LFOF LFOP LFOA KURVE  MODUS      KSV
 0    150    0  170 TRIAN  RUN        00000011
 1   7000  128   20 TRIAN  RESET      00000011
 2      0    0    0 TRIAN  RESET      00000011
 3      0    0    0 TRIAN  RESET      00000000
 4      0    0    0 TRIAN  RESET      00000000
 5      0    0    0 TRIAN  RESET      00000000
 6      0    0    0 TRIAN  RESET      00000000
                                      00000000
  A    D    S    R  EGA  FORM  MODUS
  0    0    0    0    0    +   RESET
```

```
STURM
----------------------------------------
AKTIVE STIMMEN: 1  2  3
SUSTAIN AKTIV

  F     PW    A  D  S  R CONTROL PORTA
C 3  0 2048  0 10  8 10 1000000     0
C 4  0 2048  0 10  8 10 1000000     0
C 5  0 2048  0 10  8 10 1000000     0

FILTF  RES   SCHALTER   MODUS  LAUT
 1000   12  **FILT1**   **LP**  10
            **FILT2**     BP
            **FILT3**   **HP**
               FILTEX    3OFF

LFO  LFOF LFOP LFOA KURVE  MODUS      KSV
 0    130    0  250 TRIAN  RUN        00000010
 1    160    0  250 TRIAN  RUN        00000100
 2    200    0  250 TRIAN  RUN        00001000
 3    110    0  250 TRIAN  RUN        00000000
 4      0    0    0 TRIAN  RESET      00000000
 5      0    0    0 TRIAN  RESET      00000000
 6      0    0    0 TRIAN  RESET      00000001
                                      00000000
  A    D    S    R  EGA  FORM  MODUS
  0    0    0    0    0    +   RESET
```

```
RADIO
----------------------------------------
AKTIVE STIMMEN: 1  2  3

  F     PW    A  D  S  R CONTROL PORTA
A#5  0 2048  0 10  3 10 0100000     7
F#4  0 2048  0 10  8 10 0000000     5
C 5  0 2048  0 10  8 10 0001010    10

FILTF  RES   SCHALTER   MODUS  LAUT
   0     0     FILT1      LP    15
               FILT2      BP
               FILT3      HP
               FILTEX    3OFF

LFO  LFOF LFOP LFOA KURVE  MODUS      KSV
 0  10000   58   70 SQUARE RUN        00001001
 1     90  128   70 TRIAN  RUN        00000010
 2    135  128   70 TRIAN  RUN        00000100
 3   4000  100   50 SQUARE RUN        00000000
 4      0    0    0 TRIAN  RESET      00000000
 5      0    0    0 TRIAN  RESET      00000000
 6      0    0    0 TRIAN  RESET      00000000
                                      00000000
  A    D    S    R  EGA  FORM  MODUS
  0    0    0    0    0    +   RESET
```

Auflistung der Beispielsounds für den Sound-Editor aus Ausgabe 7/85

Namen versehen. Im Untermenü »Disk« kann man dann eine solche Sound-Bank abspeichern. Um einen Sound von einer Bank in eine andere zu übertragen, lädt man zuerst die Quell-Bank, wählt den gewünschten Sound an und wählt dann die Ziel-Bank. Durch diesen Vorgang geht der angewählte Sound nicht verloren und kann dann in der Ziel-Bank abgelegt werden.

Beispielsounds

Die folgenden zehn Beispiele sollen einen Einblick geben, welche Klänge man dem SID entlocken kann. Die Parametersätze sind in einer Form abgedruckt, die an die Erscheinungsweise in den einzelnen Untermenüs angelehnt ist. Bei den Parametergruppen SCHALTER und MODUS im Filter-Menü bedeuten Sternchen, daß das entsprechende Bit aktiviert ist (reverse Darstellung auf dem Bildschirm).

Beispiel 1: MOLL WEIT

Eine relativ einfache Einstellung, bei der alle drei Stimmen in einem Moll-Akkord zusammenklingen. »Weit« bedeutet, daß die Einzeltöne nicht innerhalb einer Oktave liegen. Die Hüllkurven unterscheiden sich in ihren Decay- und Release-Werten. Dadurch klingt der tiefste Ton am längsten nach. Ein ganz leichtes Vibrato (LFOA = 1) durch LFO 0 verleiht dem Klang Leben. Man erhöhe einmal die Grundfrequenz von Stimme 2 von D # auf E4, um einen Dur-Akkord zu erhalten.

Beispiel 2: STRINGS

Dieser Sound klingt durch reichhaltigen Modulationseinsatz sehr voll, fast orchestral (im Rahmen der SID-Qualität). Es wurde dreimal die Rechteckkurve gewählt, die durch PW-Modulation räumliche Fülle erhält. Die lange Attack-Zeit (A = 5) in der Hüllkurve von Stimme 2 bewirkt ein verzögertes Einsetzen. Wenn man bei dieser Einstellung staccato (das heißt abgehackt) spielt, bleibt Stimme 2 fast unhörbar. LFO 0 und 1 sorgen für Vibrato (Frequenzmodulation). LFO 0 moduliert Stimme 1, LFO 1 Stimme 2. Stimme 3 wird von beiden LFOs moduliert. Dabei überlagern sich die Kurvenformen der beiden LFOs und bilden eine Schwebung, die sich im Auf- und Abschwellen des Vibrators in Stimme 3 (fast aufdringlich) bemerkbar macht. Dieser Effekt soll an eine Solo-Violine im Orchester erinnern. Die LFOs 2 und 3 sorgen durch PW-Modulation für noch mehr Klangfülle.

Beispiel 3: KLICK-WAH

Eine Kombination zweier Effekte. Die auf einen hohen Grundton eingestellte Stimme 2 sorgt durch einen schnellen Attack-Decay-Hüllkurvenverlauf (mit Sustain = 0) für den Klick-Effekt. Stimme 1 wird durch einen Tiefpaß mit großer Resonanz (RES = 14) geschickt. Der Soft-EG steuert über die Fil-

```
SEQUENCER
------------------------------------------
AKTIVE STIMMEN: 1  2  3

  F      PW   A  D  S  R CONTROL PORTA
C 3  0 2048  0 10  8 10 0010000    0
C 4  0 2048  0 10  8 10 0100000    0
C 5  0 2048  0 10  8 10 0100000    0

FILTF  RES   SCHALTER   MODUS  LAUT
   0    0      FILT1      LP    15
               FILT2      BP
               FILT3      HP
               FILTEX    3OFF

LFO  LFOF LFOP LFOA KURVE  MODUS     KSV
 0    800  128   88 SAWDWN RUN       00001100
 1   1600  128   44 SAWUP  RUN       00000011
 2   3200  128   22 SAWDWN RUN       00001111
 3   6400  128   11 SAWUP  RUN       00010000
 4    400    0  107 TRIAN  RUN       00010000
 5      0    0    0 TRIAN  RESET     00010000
 6      0    0    0 TRIAN  RESET     00000000
                                     00000000
  A    D    S    R  EGA  FORM  MODUS
  0    0    0    0    0    +   RESET


GEJAMMER
------------------------------------------
AKTIVE STIMMEN: 1  2  3
SOFT-EG-KOPPLUNG AKTIV

  F      PW   A  D  S  R CONTROL PORTA
C 3  0 2048  0 10  8 10 0010000    0
C 4  0 1850  0 10  8 10 0100000    0
C 5  0 2048  0 10  8 10 0100000    0

FILTF  RES   SCHALTER   MODUS  LAUT
 500   13   **FILT1**     LP     8
            **FILT2**   **BP**
            **FILT3**     HP
               FILTEX    3OFF

LFO  LFOF LFOP LFOA KURVE  MODUS     KSV
 0    400    0   30 TRIAN  RUN       10000011
 1   1400    0   20 TRIAN  RUN       10000011
 2      0    0    0 TRIAN  RESET     10000011
 3      0    0    0 TRIAN  RESET     00000000
 4      0    0    0 TRIAN  RESET     00000000
 5      0    0    0 TRIAN  RESET     00000000
 6      0    0    0 TRIAN  RESET     10000000
                                     10000000
  A    D    S    R  EGA  FORM  MODUS
 40   10    0   10  110    +   RUN


MIKROCHIP
------------------------------------------
AKTIVE STIMMEN: 1  2  3

  F      PW   A  D  S  R CONTROL PORTA
C 3  0 2048  0 10  8 10 0001010    0
G 5  0 2048  0 10  8 10 0001010    0
D 5  0 2048  0 10  8 10 0001010    0

FILTF  RES   SCHALTER   MODUS  LAUT
   0    0      FILT1      LP    15
               FILT2      BP
               FILT3      HP
               FILTEX    3OFF

LFO  LFOF LFOP LFOA KURVE  MODUS     KSV
 0  11000  120  130 SQUARE RUN       00011111
 1   5000  120   60 SQUARE RUN       00011000
 2   3000  120   80 SQUARE RUN       00011111
 3    190  120   60 SQUARE RUN       00000000
 4     70  120   30 SQUARE RUN       00000000
 5      0    0    0 TRIAN  RESET     00000000
 6      0    0    0 TRIAN  RESET     00000000
                                     00000000
  A    D    S    R  EGA  FORM  MODUS
  0    0    0    0    0    +   RESET


BULLDOZER
------------------------------------------
AKTIVE STIMMEN: 1  2  3
SUSTAIN AKTIV

  F      PW   A  D  S  R CONTROL PORTA
G 0 -1 1500  0 10  8 10 0101000    2
C 4  0 2048  0 10  8 10 1000000    2
G 1 19 2048  0 10  8 10 0010000    2

FILTF  RES   SCHALTER   MODUS  LAUT
 500   15      FILT1      LP    15
            **FILT2**   **BP**
               FILT3      HP
               FILTEX    3OFF

LFO  LFOF LFOP LFOA KURVE  MODUS     KSV
 0   8600    0  130 SAWDWN RUN       10000010
 1   8600    0  100 TRIAN  RUN       00000000
 2    200    0   50 TRIAN  RUN       10000000
 3      0    0    0 TRIAN  RESET     00000000
 4      0    0    0 TRIAN  RESET     00000000
 5      0    0    0 TRIAN  RESET     00000000
 6      0    0    0 TRIAN  RESET     00000101
                                     00000000
  A    D    S    R  EGA  FORM  MODUS
  2    2  240    4  120    -   RUN
```

Auflistung der Beispielsounds (Schluß)

terfrequenz das Öffnen und Schließen des Filters. Bei der angegebenen Parametrisierung wird das Resultat lautmalerisch besser durch »au« als durch »wah« beschrieben. Man sollte hier unbedingt einmal mit der Einstellung des Soft-EG experimentieren. Durch Verlängerung der Attack-Zeit erhält man ein »wahhau«, durch Spiegelung der Hüllkurve (FORM = -) kann man ein »auahh« erzielen. Bei diesen Versuchen sollte man auch hin und wieder die Filterfrequenz variieren.

Beispiel 4: SIRENE

Hier wird die Wirkung einer starken Frequenzmodulation demonstriert. Die Frequenzen der drei Stimmen sind so eingestellt, daß sie einen »schräg« klingenden Akkord bilden. Alle drei Stimmen werden gleichermaßen langsam und tief von LFO 0 moduliert. Hier sollte man einmal unterschiedliche LFO-Kurvenformen und LFO-Frequenzen ausprobieren. Die Zuschaltung von LFO 1 (MODUS auf RUN setzen) bewirkt durch Überlagerung einer schwächeren aber schnellen Modulation einen dramatischen Effekt.

Beispiel 5: STURM

Alle drei Stimmen erzeugen Rauschen. Die Stimmen werden wie bei der Sirene langsam und tief moduliert, allerdings jede durch einen eigenen LFO. Da die LFOs in ihrer Frequenz unterschiedlich eingestellt sind, erhalten die drei Rauschquellen unabhängig voneinander Lebendigkeit. Darüber hinaus sorgt das eigens durch LFO 0 modulierte Filter für klangliche Abwechslung. Das Filter ist hier übrigens als sogenannte Bandsperre geschaltet (LP und HP aktiv).

Beispiel 6: RADIO

Dieser Effekt soll an das Pfeifen erinnern, das beim Durchstimmen eines Kurzwellenempfängers entsteht. Des weiteren ist ein Funkfernschreib-Signal zu hören (schnelle Folge hoher Töne). Das Durchstimm-Geräusch liefert Stimme 3, die durch Stimme 2 ringmoduliert wird. Stimme 2 ist selbst nicht hörbar, beeinflußt aber wesentlich den Klang von Stimme 3. Beide Stimmen werden sehr langsam und mit unterschiedlicher Frequenz moduliert. Die LFOs 0 und 3 erzeugen dagegen schnelle Rechtecksignale, die, zu einer Treppenkurve überlagert, Stimme 1 zu einer schnellen Folge hoher Töne anregen.

Beispiel 7: SEQUENCER

Ein Sequencer ist ein Gerät, das eine vorprogrammierte Tonfolge wiederholt abspielt. Durch Modulation mit Treppenkurven kann man sequencerartige Effekte erreichen. Treppenkurven erhält man durch Überlagerung von Rechteck- und Sägezahnkurven. Die Skizze (Bild 1) gibt Beispiele dafür. Wenn man Sägezahnkurven einsetzt, muß man ein besonderes Augenmerk auf die Amplitudenverhältnisse richten, damit die Treppenstufen »gerade« bleiben.

Das vorliegende Beispiel klingt am besten, wenn man zunächst die Grundeinstellung anwählt und dann den vorher abgespeicherten Parametersatz. Dadurch werden zunächst alle LFOs in den RESET-Zustand versetzt und anschließend synchron gestartet. Sie laufen dann mit einer wohldefinierten Phasenbeziehung zueinander ab.

Beispiel 8: GEJAMMER

Hier moduliert der Soft-EG die Frequenz aller drei Stimmen. Jeder Tastendruck erzeugt dadurch ein Aufheulen, das an Katzengejammer erinnert. Der Effekt wird noch dadurch verstärkt, daß der Soft-EG auch das Filter und die Lautstärke moduliert. Eine weitere Frequenzmodulation durch zwei LFOs verleiht dem Ton nach der Decay-Phase des Soft-EG einen zusätzlichen klagenden Charakter.

Beispiel 9: MIKROCHIP

5 LFOs erzeugen durch Überlagerung Treppenkurven mit quasi-zufälligem Verlauf. Die dadurch schnell und tief modulierten Stimmen werden noch zusätzlich gegenseitig ringmoduliert. Dadurch bekommt der Gesamtklang einen metallischen Charakter. Am besten klingt diese Einstellung bei hohen Tönen.

Beispiel 10: BULLDOZER

Hier werden Motorengeräusche simuliert. Stimme 3 erzeugt ein Brummen als Grundlage. Stimme 2 erzeugt über das Filter Rauschen. Das Filter wird in schneller Folge durch LFO 0 moduliert. Dadurch wird aus dem Rauschen das Stampfen eines schweren Dieselmotors. LFO 2 moduliert ebenfalls das Filter, allerdings mit sehr niedriger Frequenz. Dadurch ändert das Diesel-Stampfen periodisch seinen Klang. Am interessantesten ist aber Stimme 1. Sie erzeugt bereits ohne Modulation ein Geräusch, das an eine schwere, trockengelaufene Stahlkette erinnert. Der Effekt kommt durch die Kombination zweier Kurvenformen (Rechteck und Dreieck) bei niedriger Oszillatorfrequenz zustande. Eine Modulation durch LFO 1 läßt die Kette allerdings noch realistischer rasseln. Drückt man die SPACE-Taste, so werden die Frequenzen von Stimme 3 (Grundgeräusch) durch den Soft-EG heruntermoduliert. Man kann sich dabei vorstellen, wie sich die schwere Maschine ächzend ins Erdreich wühlt.

(Thomas Krätzig/ev)



Logeleien (Teil 2)

Was hat es mit dem AND, dem OR und dem selten benutzten WAIT im Basic auf sich? Das und noch einiges mehr wird uns in dieser zweiten Folge der Logeleien beschäftigen.

In der täglichen Umgangssprache verwenden wir oft das Wörtchen »und«. Häufig wird es dabei genauso gebraucht wie ein logisches AND (siehe Bild 1).

Wir sehen zwei Aussagen, die durch AND zur Gesamtaussage verknüpft wurden. Unser Gefühl sagt uns, daß die Gesamtaussage falsch ist, obwohl eine von den Einzelaussagen richtig sein kann. Sind beide Aussagen falsch (wenn A = 100), dann ist auch die Gesamtaussage falsch. Nur wenn beide Aussagen richtig wären, wäre auch die Gesamtaussage wahr. Sehen wir uns dazu die Wahrheitstabelle an (Tabelle 1).

A1	A2	A1 AND A2
W	W	W
W	F	F
F	W	F
F	F	F

Tabelle 1. Die Wahrheit kommt ans Licht. Wahrheitstabelle Typ 1 am Beispiel der AND-Verknüpfung.

In den linken Spalten stehen untereinander alle möglichen Kombinationen der Wahrheitswerte von Aussage 1 (A1) und Aussage 2 (A2). Die rechte Spalte zeigt das Ergebnis der AND-Verknüpfung, das wir aus unserem Textbeispiel gewonnen haben.

Eine andere — ebenfalls bei der Verknüpfung von nur zwei Aussagen häufig gebrauchte — Darstellungsform einer Wahrheitstabelle zeigt Ihnen die Tabelle 2.

Hier werden in der linken Spalte die möglichen Wahrheitswerte der einen, in der Kopfzeile die der anderen Aussage eingetragen. Kreuzweise verknüpft man diese dann (hier also durch die AND-Verknüpfung) miteinander, was die quadratische Matrix der Wahrheitswerte der Gesamtaussage ergibt. Welche von beiden Formen Sie verwenden, ist Ihrem Geschmack überlassen. Hat man allerdings mehr als zwei Aussagen zu verknüpfen, dann ist die erste Form überschaubarer.

A1↓ A2→	W	F
W	W	F
F	F	F

Tabelle 2. Typ 2 der Wahrheitstabelle bei der AND-Verknüpfung

Wie kann man AND mit Aussagen nutzen? Sehen wir uns zunächst ein einfaches Beispiel an:

```
10 INPUT C
20 A=(C>5):B=(C<10)
30 IF A AND B THEN
PRINT"C LIEGT ZWISCHEN
5 UND 10":GOTO 50
40 PRINT"C IST KLEINER/
GLEICH 5 ODER GROES-
SER/GLEICH 10"
50 END
```

Aus diesem simplen Beispiel kann man sehen, daß sich mittels AND-verknüpfter Aussagen eine Klassifizierung vornehmen läßt. Häufig stellt sich folgendes Problem: Eine große Anzahl durch irgendwelche Peripherie eingehender Werte (zum Beispiel Meßwerte von Rausch-Frequenzen oder ähnliche) soll in Klassen einsortiert werden, um beispielsweise eine Häufigkeits-Verteilung zu erkennen. Der gesamte mögliche Frequenzbereich wird dann in sogenannte Klassengrenzwerte unterteilt, die man in ein Array A(N+1) einliest. Jeder eingehende Meßwert X wird dann klassifiziert, indem er die folgende Programmzeile durchläuft:

```
FOR I=1 TO N:A=(X>A(I)):
B=(X<A(I+1)): IF A AND B
THEN Z(I)=Z(I)+1
```

Dabei ergibt sich im Array Z(N) schließlich die Häufigkeitsverteilung.

Zahlen mit AND verknüpfen

Ebenso wie bei NOT muß die AND-Verknüpfung von Zahlen wieder auf der Bit-Ebene beobachtet werden. Dementsprechend setzt sich die Wahrheitstabelle wieder aus den Binärziffern 0 und 1 anstelle der Wahrheitswerte F und W zusammen (siehe Tabelle 3).

A1	A2	A1 AND A2
1	1	1
1	0	0
0	1	0
0	0	0

Tabelle 3. Wahrheitstabelle mit Zahlen der AND-Operation

Falls Ihnen die andere Form der Tabelle besser gefällt, finden Sie diese in Tabelle 4.

A1↓ A2→	1	0
1	1	0
0	0	0

Tabelle 4. Der zweite Typ der Wahrheitstabelle bei der AND-Verknüpfung von Zahlen

Nur wenn beide miteinander AND-verknüpften Bits den Wert 1 haben, ist auch das Ergebnis 1. Nun können Sie ein wenig probieren — mit Hilfe des in der letzten Folge gezeigten Hilfsprogrammes — welche Möglichkeiten die AND-Operation bietet.

Wie Sie bei Ihren Versuchen vielleicht bemerkt haben, eignet sich AND vor allem zum Löschen von Bits. Probieren Sie mal das folgende Beispiel aus:

1. Zahl: 255
2. Zahl: 4

Im Ergebnis sehen Sie, daß alle Bits der ersten Zahl gelöscht wurden, außer dem Bit 2 (nur dort wurden zwei Bits mit dem Wert 1 AND-verknüpft). Die zweite Zahl nennt man eine Maske. So eine Maske konstruiert man ganz gezielt für das Löschen von Bits durch eine AND-Verknüpfung. Man nennt sie daher auch eine »AND-Maske«. Wollen Sie also erreichen, daß in einer beliebigen vorhandenen Zahl X alle Bits bis auf das zweite gelöscht werden (wobei vorausgesetzt wird, daß überhaupt Bit 2 von X gesetzt ist), dann AND-verknüpfen Sie diese Zahl mit der AND-Maske 4.

Zur Frage der Anwendung: Es gibt im Commodore 64 eine ganze Anzahl sogenannter Kontrollregister, in denen jedes Bit eine bestimmte Rolle spielt. Beispielsweise steuert das Register 53269 das Ein- oder Ausschalten von Sprites. Soll von mehreren abgebildeten Sprites nun Sprite 2 abgeschaltet werden, dann erfordert das ein gezieltes Löschen des Bit 2 dieses Registers. Alle anderen Bits sollen unberührt bleiben. Die Maske muß also außer bei Bit 2 (das den Wert 0 haben muß) überall eine 1 enthalten:

1111 1011

Das ist die Dezimalzahl 251 (oder 255-4). Der Basic-Befehl lautet daher:

```
POKE53269,PEEK(53269)
AND251
```

oder allgemein:

```
POKE53269,PEEK(53269)
AND(255-N)
```

Dabei ist N die Nummer des zu löschenden Sprites. Sollen mehrere Bits gleichzeitig gelöscht werden, müssen in der AND-Maske einfach an den entsprechenden Stellen Nullen auftreten. Das dürfte nun kein Problem mehr für Sie sein.

Der Vollständigkeit halber sei abschließend noch erwähnt, daß in der Literatur

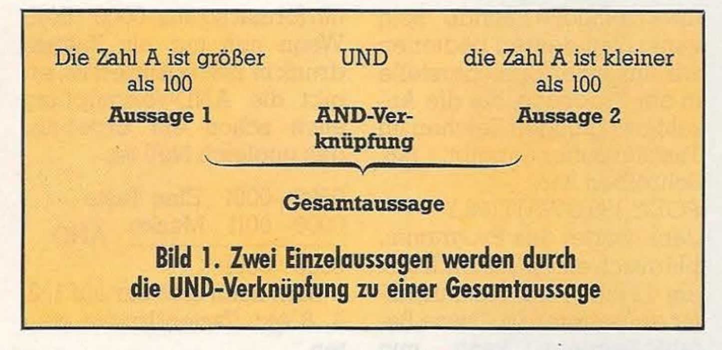


Bild 1. Zwei Einzelaussagen werden durch die UND-Verknüpfung zu einer Gesamtaussage

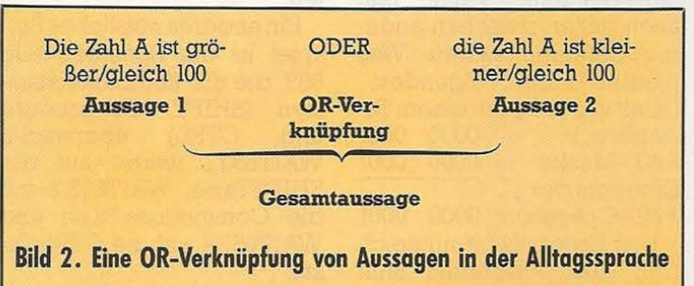


Bild 2. Eine OR-Verknüpfung von Aussagen in der Alltagssprache

A1	A2	A1 OR A2
W	W	W
W	F	W
F	W	W
F	F	F

Tabelle 5. Wahrheitstabelle der OR-Verknüpfung von Aussagen

A1	A2	A1 OR A2
1	1	1
1	0	1
0	1	1
0	0	0

Tabelle 6. So werden Zahlen bitweise OR-verknüpft

für die AND-Verknüpfung manchmal der Begriff »Konjunktion« verwendet wird.

Auch hier soll uns ein umgangssprachliches Beispiel helfen (siehe Bild 2).

Die Gesamtaussage ist richtig, wenn mindestens eine der beiden Einzelaussagen wahr ist. Die Wahrheitstabelle verdeutlicht diese sogenannte OR-Verknüpfung (siehe Tabelle 5).

Vielleicht ist Ihnen auch schon ein kleines Sprachproblem aufgefallen: Das Wort »oder« wird in zwei verschiedenen Bedeutungen gebraucht. Man unterscheidet:

1) Inklusiv-Oder

Das ist das Oder, welches in diesem Abschnitt betrachtet wird, die OR-Verknüpfung.

2) Exklusiv-Oder

Da haben wir ein anderes Oder, das wir noch untersuchen werden. In der normalen Sprache kann man es am besten durch »entweder...oder« umschreiben. In der mathematischen Logik und in Programmierer-Kreisen nennt man es EOR oder manchmal auch XOR.

OR-verknüpfte Aussagen finden gerne Anwendung bei Menüabfragen. Es sei beispielsweise ein Menü mit fünf Optionen gegeben, das mittels GET abgefragt werde. Dann kann die entsprechende Zeile lauten:

```
100 GET A:IF A<1 OR A>5
THEN 100
```

Für viele von Ihnen ist das vermutlich ein alter Hut. Aber es gibt vielleicht doch auch einige Leser, die noch verwenden:

```
100 GET A:IF A="" THEN
100
110 IF A = 1 THEN...
120 IF A = 2 THEN...
160 GOTO 100
```

Zahlen OR-verknüpfen

Zum OR-Verknüpfen von Zahlen steigen wir wieder in die Bit-Ebene herab. Hier zunächst einmal die Wertetabelle (Tabelle 6).

Nur dort also, wo beide miteinander OR-verknüpften Bits gleich Null sind, wird auch das Ergebnis eine Null. Gewissermaßen haben wir hier das Gegenteil der AND-Verknüpfung vorliegen.

Das drückt sich auch in der Anwendung aus: Die OR-Operation kann zum gezielten Setzen von Bits verwendet werden. Auch das probieren Sie am besten mittels des Hilfsprogrammes aus. Gleichgültig, welche erste Zahl Sie angeben: Wenn Sie als zweite Zahl eine 4 (also 0000 0100) damit OR-verknüpfen, wird das Bit 2 des Ergebnisses auf 1 gesetzt sein. Alle anderen Bits bleiben unverändert erhalten. Die zweite Zahl ist hier wieder die Maske. Weil die Maske OR-verknüpft wird, spricht man von einer OR-Maske.

OR gibt uns die Möglichkeit, einzelne Bits in den Kontrollregistern zu setzen. So könnte man das vorhin (bei der Erklärung der AND-Operation) abgeschaltete Sprite wieder einschalten durch:

```
POKE53269,PEEK(53269)
OR4
```

In der Literatur wird die OR-Operation häufig auch als »Disjunktion« bezeichnet.

Das exklusive Oder und Aussagen

NOT, AND und OR sind im Basic-Sprachvorrat enthalten. Die EOR-Operation ist in versteckter Form verfügbar und zwar im WAIT-Befehl. Bevor wir uns aber diesem zuwenden, soll erst einmal die EOR-Verknüpfung untersucht werden. Wir hatten vorhin schon angedeutet, daß EOR umgangssprachlich durch »entweder...oder« umschrieben werden kann. Ein Beispiel ist auch in Bild 3 zu sehen.



Bild 3. Ein Sprichwort bringt es an den Tag: eine Exklusiv-Oder-Verknüpfung

Immer dann, wenn nur eine der beiden Teilaussagen wahr ist, ist auch die Gesamtaussage richtig. Die Wahrheitstabelle zur EOR-Verknüpfung finden Sie als Tabelle 7. Sehen wir uns nun noch an, was die EOR-Operation mit Zahlen anrichtet.

A1	A2	A1 EOR A2
W	W	F
W	F	W
F	W	W
F	F	F

Tabelle 7. Aussagen exklusiv-oder verknüpft. Die Wahrheitstabelle

A1	A2	A1 EOR A2
1	1	0
1	0	1
0	1	1
0	0	0

Tabelle 8. Die EOR-Operation auf der Bit-Ebene

Zahlen exklusiv »geODERt«

In unserem Hilfsprogramm ist auch diese Möglichkeit vorgesehen. Sie erkennen, wenn Sie die Option 4 einmal ausprobieren, daß die EOR-Operation bei zwei gleichen Bits immer 0, bei ungleichen immer 1 ergibt.

Die Wahrheitstabelle sieht daher so aus, wie in Tabelle 8 gezeigt.

Ein Fakt ist an der EOR-Operation interessant: Wendet man auf eine beliebige Zahl zweimal hintereinander dieselbe EOR-Maske an (muß ich nun sicher nicht mehr erklären, oder?), dann ist das Ergebnis wieder die Ausgangszahl. Probieren Sie es mal aus, zum Beispiel mit:

```
        234
1. Mal   56
        ---EOR
        210
        210
2. Mal   56
        ---EOR
        234
```

Damit haben wir die Voraussetzungen erfüllt, den WAIT-Befehl zu verstehen.

WAIT: Ein Aschenputtel in Basic

Wie es schon der Name sagt (wait=warten), hält WAIT ein Programm so lange an, bis in einer spezifizierten Speicherstelle ein bestimmtes Bit-Muster aufgetreten ist. Dieses Bit-Muster gibt man durch zwei Masken vor, von denen die erste (obligatorische) eine AND-, die zweite eine EOR-Maske ist. So sieht die Syntax des WAIT-Befehls aus:
WAIT Speicherstelle, AND-Maske, EOR-Maske

Der effektive Einsatz von WAIT erfordert eine recht gute Kenntnis der Speicherbelegung unseres Commodore 64, was vermutlich einer der Gründe für die seltene Benutzung dieses Befehls ist.

Wie eben schon kurz gesagt, kann WAIT mit einem oder mit zwei Argumenten betrieben werden. Sehen wir uns zunächst mal an, was mit nur einem Argument möglich ist.

Hier wird der in der adressierten Speicherstelle enthaltene Wert mit der AND-Maske verknüpft. Ergibt sich dabei ein Wert, der ungleich Null ist, dann fährt das Programm fort. Im anderen Fall wartet es, bis der Speicherinhalt dieser Anforderung entspricht (also bis irgendwann einmal die AND-Verknüpfung des Speicherinhaltes mit der AND-Maske keine Null mehr ergibt) oder aber bis man, des Wartens müde, das Programm durch RUN/STOP-RESTORE unterbricht. Sehen wir uns dazu ein praktisches Beispiel an. Vielfach werden Programme bis zu einem Tastendruck mittels:
100 GET A$:IFA$="''THEN 100
angehalten. Das hat den Nachteil, daß man dafür eine Basic-Zeile vergeben muß, was beispielsweise beim VC 20 in der Grundversion eine unverzeihliche Sünde sein kann. Statt dessen bedienen wir uns einer Speicherstelle in der Zeropage, die die Anzahl der gültigen Zeichen im Tastaturpuffer angibt: 198. Schreiben wir:
POKE 198,0:WAIT198,1
dann wartet das Programm, bis durch einen Tastendruck ein Zeichen im Tastaturpuffer aufgetreten ist. Diese Befehls-Sequenz kann nun auch mitten zwischen anderen Befehlen stehen. Was passiert dabei? Folgendes:

Inhalt von 198 nach einem Tastendruck: 0000 0001
AND-Maske: 0000 0001
Ergebnis der AND-Operation: 0000 0001

Das Ergebnis ist ungleich Null, das Programm läuft weiter.

Wenn wir WAIT198,2 eingeben, wartet das Programm auf zwei Tastendrucke. Was geschieht bei WAIT 198,3? Probieren Sie es aus: Schon nach dem ersten Tastendruck läuft das Programm weiter. Sehen wir uns mal an, weshalb. 3 sieht im Binärformat so aus: 0000 0011. Wenn nun nur ein Tastendruck in 198 registriert ist, ergibt die AND-Verknüpfung auch schon ein Ergebnis, das ungleich Null ist:

0000 0001 Eine Taste
0000 0011 Maske AND
0000 0001

Man kann also nur auf 1, 2, 4, 8 etc. Tastendrucke warten.

Ein anderes nützliches Beispiel ist die Speicherstelle 653, die die Kombinationstasten (SHIFT, Commodore und CTRL) überwacht. WAIT653,1 wartet auf die SHIFT-Taste, WAIT653,2 auf die Commodore-Taste und WAIT653,4 auf die CTRL-Taste.

Damit wollen wir für diesmal aufhören zu »logeln«. In der nächsten Folge geht es dann um den WAIT-Befehl mit zwei Argumenten. Außerdem werden wir feststellen, ob ein Computer auch Schlußfolgerungen ziehen kann.

(Heimo Ponnath/gk)

Sortieren mit dem Computer (Teil 4)

Quicksort verdient seinen Namen zu Recht. Aber er kann auch zu einem ausgesprochenen Langweiler werden. Sogar Bubblesort kann schneller sein als alle anderen Sortierroutinen. Woran das liegt, erfahren Sie im folgenden Artikel.

Der Quicksort-Algorithmus wurde bereits 1962 von R. Hoare entwickelt und ist trotzdem bis heute die schnellste Methode zum Sortieren großer, zufallsbesetzter Felder geblieben. Der Deutlichkeit halber betone ich noch einmal zufallsbesetzt, denn nur bei dieser Art der Verteilung der Feldelemente kann Quicksort wirklich effektiv arbeiten. Haben wir ein Array, das zum Beispiel absteigend sortiert ist (9, 8, 7, ..., 1) und sollen hier nur wenige Elemente neu einsortiert werden, so wird Quicksort im wahrsten Sinne des Wortes zum »Slowsort« und benötigt eine ewige Zeit zur Bearbeitung des Feldes.

Andererseits lohnt sich der Einsatz von Quicksort auch dann nicht, wenn es darum geht, beispielsweise eine Kartei zu führen, bei der laufend neue Elemente in ein schon vorsortiertes Feld eingefügt werden sollen (zum Beispiel der Buchstabe D in das Feld A, B, C, F, H, M, N, O, ...). Hier eignet sich unser (normalerweise langsamster) Bubblesort-2-Algorithmus sehr gut; in diesem Fall findet nämlich nur ein einziger Durchlauf statt, bevor der Algorithmus beendet wird (wie Sie wissen, verwendet Bubblesort 2 eine zusätzliche Variable, die anzeigt, ob noch weitere Sortierdurchläufe notwendig sind).

Bei der Effektivität unserer Sortierprogramme sollen uns diese Spezialanwendungen jedoch nicht weiter interessieren. Hier geht es nur um das Problem, ein völlig »durcheinandergeratenes« Feld wieder in Ordnung zu bringen, und das bitte möglichst schnell.

Wie funktioniert Quicksort?

So wollen wir uns nun einmal den Quicksort-Algorithmus etwas genauer betrachten. Er ist nicht im mindesten mit den bisherigen Methoden zu vergleichen, und wunderbarerweise zählt er trotz seiner Leistungsfähigkeit zu den etwas leichter verständlichen Algorithmen.

Das Prinzip von Quicksort ist folgendes:

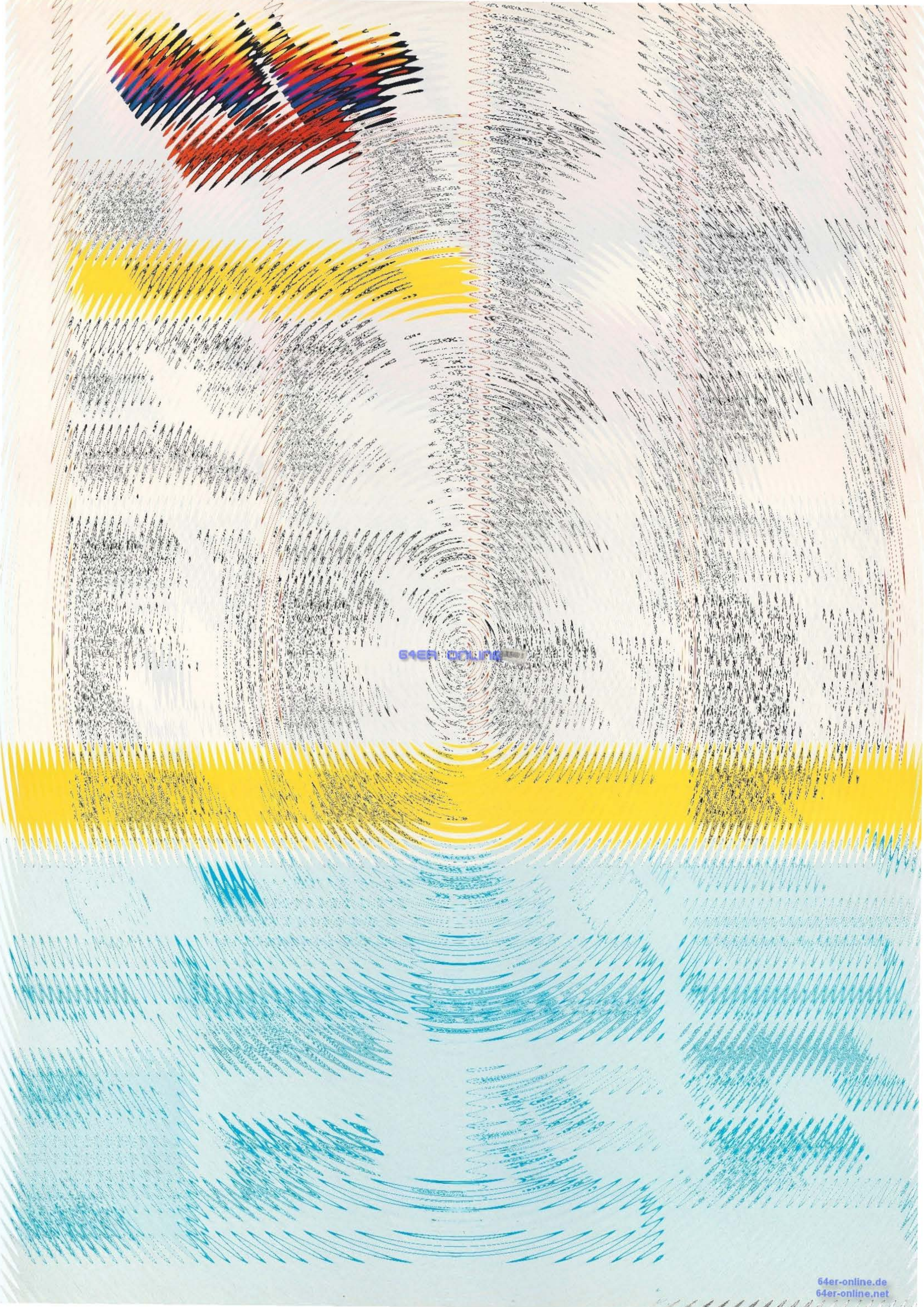
Zuallererst wird aus dem gesamten Variablenfeld ein (beliebiges) Element herausgegriffen und »beiseite gelegt«. Dieses Element dient nun als Vergleichswert für das gesamte übrige Array. Jetzt werden alle Elemente, die kleiner als das Vergleichselement sind, links (also auf niedrigere Positionen) und alle Elemente, die größer sind, rechts (also auf höhere Positionen) vom Vergleichselement abgespeichert. Wir erhalten so ein Variablenfeld, bei dem alle kleineren Elemente oberhalb und alle größeren Elemente unterhalb des Vergleichselements stehen. Bild 1 zeigt noch einmal genau, was gemeint ist.

Nachdem diese »Gesamtsortierung« vollzogen wurde, haben wir ein schon sehr grob sortiertes Feld vorliegen. Nun teilen wir die beiden neuen Teilfelder (oberhalb und unterhalb des Vergleichselements) wiederum durch jeweils ein neues Vergleichselement auf, wobei diese beiden Teilfelder auf die oben beschriebene Art erneut sortiert werden. Als Ergebnis haben wir dann vier teilsortierte Feldabschnitte vorliegen, die wiederum, jedes für sich, geteilt und sortiert werden.

Setzen wir diese Teilungen immer weiter fort, so werden wir irgendwann einmal die Teilfeldlänge 1 erreichen, womit unser Gesamtfeld fertig sortiert wäre.

Wie Sie sicherlich bemerken, ist die Effektivität dieses Algorithmus natürlich stark von der Wahl des jeweiligen Vergleichselements abhängig. Optimal wäre jedesmal ein Vergleichselement, das etwa das Mittel der zugehörigen Teilliste ausmacht und diese so in zwei gleich große Abschnitte trennt.





Die Suche nach einem solchen optimalen Vergleichselement würde sich jedoch so aufwendig gestalten, daß die Geschwindigkeit von Quicksort stark darunter leiden müßte. Aus diesem Grund geht man bei der Wahl des betreffenden Elements einen anderen Weg. Es wird einfach jedesmal das Element gewählt, das genau in der Mitte der Teilliste steht. Auf diese Weise erhält man bei zufallsbesetzten Feldern zwar auch sehr ungünstige Werte; dieser Mangel wird jedoch durch eine ebensogroße Zahl von extrem günstigen Werten ausgeglichen.

03	05	23	01	17	88	47	33	21	14
03	05	14	01	17	88	47	33	21	23
03	05	01	14	17	47	33	21	23	88
01	03	05	14	17	21	23	33	47	88
01	03	05	14	17	21	23	33	47	88

Bild 1. Sortierbeispiel von Quicksort

Im Geschwindigkeitstest zeigt dieser Quicksort-Algorithmus dann auch seine verblüffenden Eigenschaften.

Bevor wir jedoch auf einen Vergleich sämtlicher bisher besprochener Sortiermethoden eingehen, noch etwas zur Programmierung von Quicksort.

Was ist rekursives Programmieren?

Während des Programmablaufs kommt ein Unterprogramm immer wieder in der gleichen Form vor und zwar das Sortieren eines Teilfeldes anhand des Vergleichselements. Dieses Unterprogramm stellt nun auch die neuen Teillisten fest und müßte bei einem sofortigen Rücksprung mit »RETURN« sämtliche neuen Parameter zwischenspeichern, um dann erneut aufgerufen zu werden ...

Diese umständliche Programmiertechnik, die für einen linearen Programmablauf sorgt, wurde bei Quicksort nicht verwendet. Hier werden vielmehr die neu festgestellten Teillisten sofort wieder bearbeitet, um danach ebenfalls wieder die nächsten Teillisten herzustellen.

Wie funktioniert nun eine solche Programmiertechnik?

Man geht davon aus, daß der eigentliche Sortier- und Teilalgorithmus in einer Unterroutine untergebracht ist, wobei diese mit GOSUB aufgerufen wird. Wurde nun die erste Teilung des Variablenfeldes durchgeführt, so werden sofort die Werte für die nächste Halbierung dieser beiden Teilfelder bereitgestellt. Danach ruft sich die Unterroutine sofort wieder **selbst** auf, um auch die neue Sortierung und Teilung ablaufen zu lassen. Dabei entstehen wieder zwei neue Teilfelder, die wiederum sofort bearbeitet werden, etc. ...

Bild 2 zeigt anschaulich, was gemeint ist.

Eine solche Programmiertechnik, bei der sich Programmteile selbst aufrufen, nennt man rekursiv!

Nun, könnten Sie jetzt einwenden, dieses Programm verschachtelt sich doch immer weiter, ohne daß ein Ende abzusehen ist. Wie findet der Computer denn aus diesem »Irrgarten« wieder heraus?

Die Sache ist relativ einfach. Irgendwann wird der Computer bei der Teillistenlänge 1 angekommen sein. Hat er das festgestellt, so erfolgt kein weiterer Selbstaufruf mehr, sondern es wird ein »RETURN« durchgeführt. Jetzt ist aber unser Unterprogramm so ausgelegt, daß es jeweils **alle** Teillisten der gleichen Länge nacheinander bearbeitet. Erfolgt also das erste RETURN, so bearbeitet der Computer die zweite Teilliste der Länge 2, bis er auch hier wieder teilt und bei 1 ankommt ...

Dieses Bearbeiten der zwei letzten Teillisten setzt sich so lange fort, bis der Computer bei der letzten Teilliste dieser Ebene angekommen ist. Hier erfolgt wiederum ein RETURN, so daß nun auf die drittletzte Ebene mit der Größe 4 zurückgesprungen wird. Auch dort werden sämtliche Elemente bearbeitet, bis wir irgendwann wieder bei einer Teilliste der Länge A (gesamtes Feld) angekommen sind, worauf der Quicksortalgorithmus verlassen wird.

Dieses Prinzip ist nicht so ohne weiteres verständlich. Am besten sehen Sie sich noch einmal Bild 2 an; dort ist der ganze rekursive Algorithmus grafisch dargestellt.

An der Geschwindigkeit von Quicksort wird deutlich, wie effektiv rekursive Programmiertechnik sein kann.

Probleme der Rekursion in Basic

Bei unserem Quicksort in Basic stellt sich jedoch noch ein zusätzliches Problem: Basic ist keine strukturierte Sprache!

Was das bedeutet, sei am Beispiel von Pascal kurz erläutert.

Bei einer strukturierten Sprache werden Unterprogramme als eigene Einheiten mit eigenen Variablen betrachtet. Es kann also vorkommen, daß eine Unterroutine die gleichen Variablen wie das übergeordnete Programm enthält, wobei diesen jedoch andere Werte zugeordnet sind.

Hat also beispielsweise das Hauptprogramm in Pascal eine Variable X mit dem Wert 3 belegt und springt nun ein Unterprogramm an, in dem diese Variable ebenfalls verwendet wird, so wird der Inhalt von X auf einen Software-Stack gerettet und erst anschließend das Unterprogramm aufgerufen. Bei der Rückkehr aus dieser Unterroutine holt der Computer den Wert von X wieder vom Stack und übergibt ihn dem Hauptprogramm.

Bei einer rekursiven Programmiertechnik bleiben also alle Parameter, die von einer Routine erarbeitet wurden, erhalten und können später, gemäß der Reihenfolge, weiter verwendet werden.

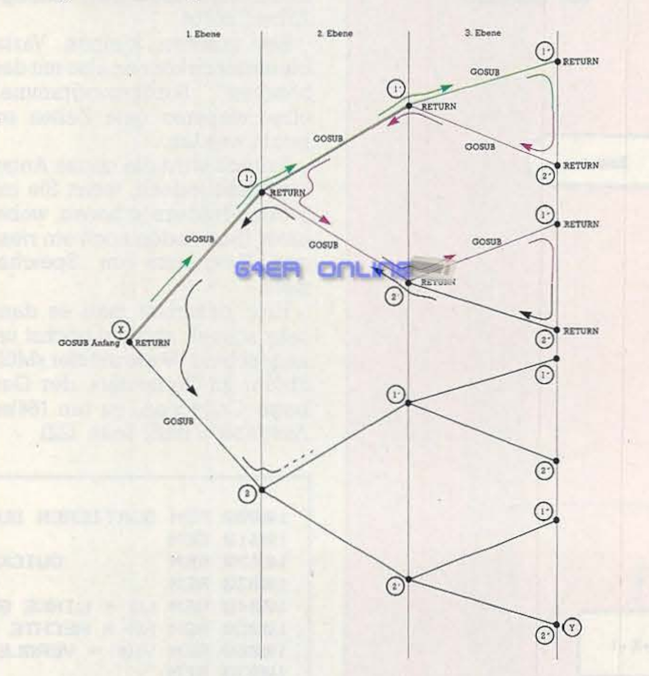
Nicht so in Basic!

Hier gibt es nur jeweils **eine** Variable gleichen Namens, deren Wert sowohl vom Hauptprogramm als auch von Unterroutinen gleichermaßen beeinflußt werden kann. Haben Sie also die Variable X mit dem Wert 3, so erfolgt bei einem Unterprogramm, in dem es heißt: X=5, ein Verlust der 3, der auch bei der Rückkehr ins Hauptprogramm nicht aufgehoben wird.

Aus diesem Grund können Sie im Basic-Quicksort noch ein weiteres Variablenfeld erkennen, in dem die wichtigen Parameter der Teilfelder abgespeichert werden, damit sie bei der Rückkehr von einer Ebene auf eine höhere wieder zur Verfügung stehen.

Wenn Sie das Prinzip der rekursiven Programmiertechnik noch nicht ganz verstanden haben, sollten Sie nicht die Mühe scheuen, noch einmal von vorne

Bild 2. Grafische Darstellung der rekursiven Programmierung bei Quicksort. Der besseren Übersicht halber sind nur drei Verschachtelungsebenen dargestellt.



Erklärung zu Bild 2:

Das rekursive Unterprogramm arbeitet immer innerhalb **eines** Aufrufes **zwei** Teilfelder ab, indem es zwischen beiden die Elemente vertauscht (Sortierung anhand des Vergleichselements).

Die rekursive Technik funktioniert nun wie ein Baum, bei dem nacheinander alle Äste abgefahren werden.

Zuerst wird das Gesamtfeld in zwei Teilfelder (entsprechend des Vergleichselements) aufsortiert (1. Ebene). Von diesen zwei Teilfeldern wird nun zuerst das erste Teilfeld wiederum aufgeteilt (grüner Pfeil X → 1). Da noch nicht Länge 1 erreicht wurde, wird das neue Teilfeld wiederum aufgeteilt (grüner Pfeil 1' → 1''). Jetzt haben die Teilfelder die Länge 1 und sind vollständig sortiert, weshalb nun auf eine höhere Ebene (RETURN bei 1'' und 2'') zurückgekehrt wird (roter Pfeil 2'' → 1' → 1).

Jetzt wird (wiederum mit GOSUB) das zweite Teilfeld der 2. Ebene (2') ebenfalls vollständig sortiert.....

Diese Vorgänge wiederholen sich, bis beim Punkt Y mit drei RETURNs wieder ganz zurückgesprungen wird, da das Unterprogramm auf 1. Ebene vollständig abgearbeitet wurde.

mit dem Lesen dieses Artikels zu beginnen. Diese Programmiermethode eignet sich nämlich zu allen algorithmischen Lösungen von Problemen mit ähnlicher Struktur, und Sie werden anhand von Quicksort erkennen können, wie effektiv und leistungsstark ein solches Programm wird.

Wie leistungsstark unsere sämtlichen Sortierprogramme sind, das soll im folgenden dargestellt werden, wobei Sie auch für Quicksort wieder einen Programmablaufplan vorfinden, der das Umschreiben auf andere Programmiersprachen erleichtern soll (Bild 3).

Nun aber zu unserem Vergleichstest.

Für diesen Test habe ich sämtliche Sortierprogramme soweit optimiert, indem ich alle REMs und alle zusätzlichen Leerzeichen entfernt habe. Außerdem wurde die dauernde Zwischenausgabe der Daten auf Bildschirm oder Drucker weggelassen, so daß sich die Zeitmessung jetzt nur auf den reinen Algorithmus bezieht.

Sortieralgorithmen im Vergleich

Getestet wurde bei allen Kandidaten ein zufallsbesetztes Feld von jeweils 100, 250 und 500 Elementen. Die Ergebnisse dieses Tests sind in Bild 4 dargestellt. Wie Sie sehen, schneidet Quicksort mit Abstand am besten ab; dicht gefolgt von Heapsort und etwas weiter von Shellsort. Diese drei Sortiermethoden eignen sich also für den praktischen Einsatz bei zufallsbesetzten Feldern, wobei im Vergleich zu den »niederen« Algorithmen sehr große Zeitvorteile zu verzeichnen sind. Bei 500 Elementen ist beispielsweise Quicksort über 16mal so schnell, wie unser (verbesserter) Bubblesort-2-Algorithmus.

Wie Sie sicherlich ahnen, hat diese schlechte Zeit, die sogar unser »normales« Bubblesort unterbietet, etwas mit der zusätzlichen Abfrage auf Vertauschungen zu tun.

Bei zufallsbesetzten Feldern ist Bubblesort 2 nicht akzeptabel!

Haben wir jedoch ein schon aufsteigend sortiertes Feld vorliegen, bei dem nur einige neue Elemente eingefügt werden sollen, so wird Bubblesort 2 fast unschlagbar, da hier die zusätzliche Abfrage für ein schnelles Ende, ohne zusätzliche, unnötige Arbeit, sorgt.

Bei unseren kleinen Variablenfeldern können also mit den höheren Sortierprogrammen ohne weiteres gute Zeiten erreicht werden.

Kritisch wird die ganze Angelegenheit jedoch, wenn Sie mit großen Feldern arbeiten, wobei unter Umständen noch ein riesiges Programm im Speicher steht.

Hier bekommt man es dann sehr schnell und auf höchst unangenehme Weise mit der »Müllabfuhr im Computer«, der Garbage Collection, zu tun (64'er, Ausgabe 1/1985, Seite 122).

Diese Tatsache ist jedoch nicht weiter verwunderlich, wenn man bedenkt, daß wir bei zufallsbesetzten Feldern beinahe das gesamte Array neu definieren und dabei entsprechend viel »Müll« erzeugen.

Sortieren ohne Müll

Haben Sie vielleicht schon einmal an eine andere Methode der Sortierung von Arrays gedacht? Es gibt nämlich noch eine weitere Möglichkeit, bei der man ohne viele Stringverschiebungen auskommt. Diese Möglichkeit der Sortierung und das Problem der Garbage Collection soll uns das nächstemal interessieren, wo wir uns dann einmal mit dem Sortieren der Indizes der Feldvariablen befassen. Außerdem können Sie sich schon einmal Gedanken zum Thema »Sortieren mehrdimensionaler Felder« machen. Das ist nämlich nicht annähernd so leicht, wie es vielleicht aussehen mag und erfordert eine ganze Menge Schweiß.

Für heute wollen wir es aber nun genug sein lassen. Experimentieren Sie doch ein wenig mit Quicksort, und lernen Sie diesen Algorithmus genauer kennen. Ich bin davon überzeugt, daß sich Ihnen früher oder später ein Problem stellt, bei dem Sie auf einen guten und schnellen Sortieralgorithmus angewiesen sein werden und den Sie nun Dank R. Hoare auch besitzen.

(Karsten Schramm/gk)

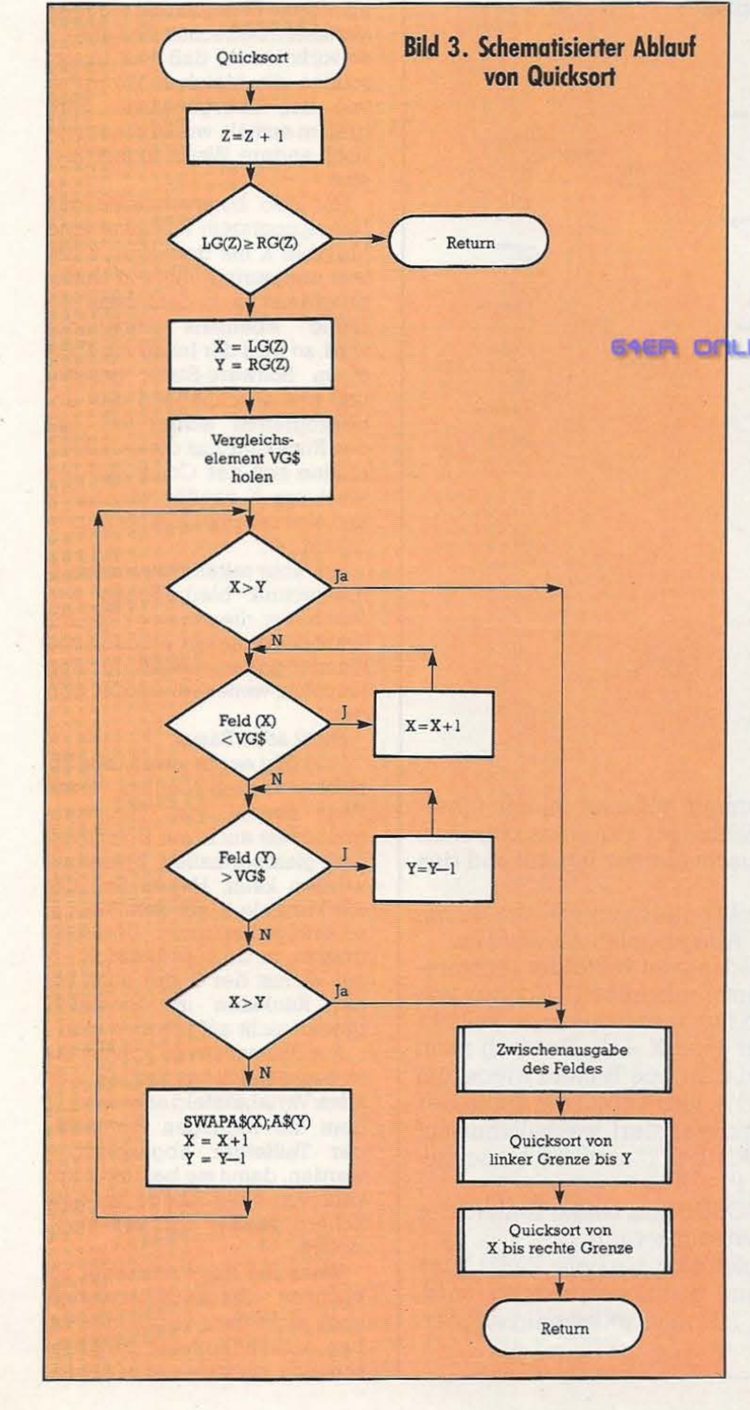
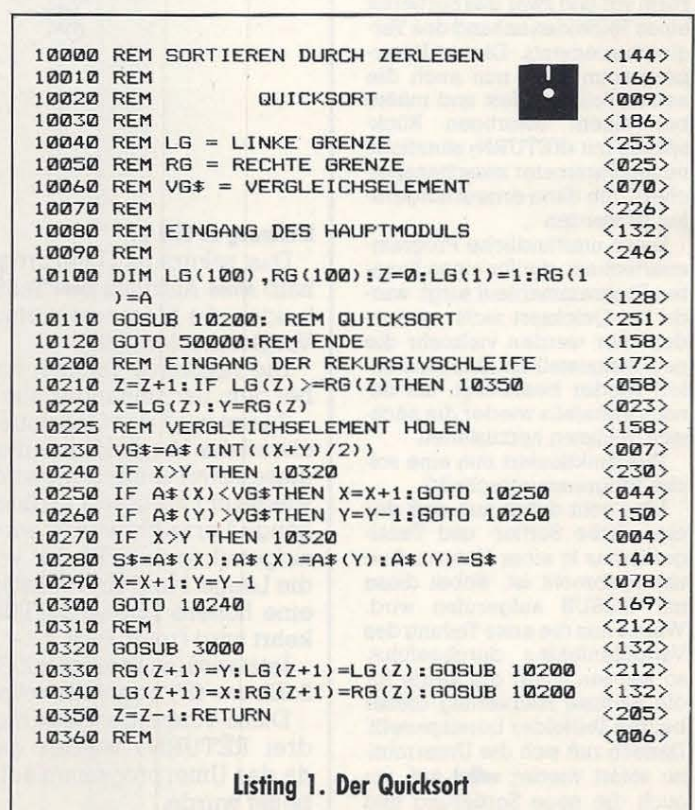


Bild 3. Schematisierter Ablauf von Quicksort



```
10000 REM SORTIEREN DURCH ZERLEGEN                 <144>
10010 REM                                          <166>
10020 REM             QUICKSORT                    <009>
10030 REM                                          <186>
10040 REM LG = LINKE GRENZE                        <253>
10050 REM RG = RECHTE GRENZE                       <025>
10060 REM VG$ = VERGLEICHSELEMENT                  <070>
10070 REM                                          <226>
10080 REM EINGANG DES HAUPTMODULS                  <132>
10090 REM                                          <246>
10100 DIM LG(100),RG(100):Z=0:LG(1)=1:RG(1
      )=A                                          <128>
10110 GOSUB 10200: REM QUICKSORT                   <251>
10120 GOTO 50000:REM ENDE                          <158>
10200 REM EINGANG DER REKURSIVSCHLEIFE             <172>
10210 Z=Z+1: IF LG(Z)>=RG(Z)THEN 10350             <058>
10220 X=LG(Z):Y=RG(Z)                              <023>
10225 REM VERGLEICHSELEMENT HOLEN                  <158>
10230 VG$=A$(INT((X+Y)/2))                         <007>
10240 IF X>Y THEN 10320                            <230>
10250 IF A$(X)<VG$THEN X=X+1:GOTO 10250            <044>
10260 IF A$(Y)>VG$THEN Y=Y-1:GOTO 10260            <150>
10270 IF X>Y THEN 10320                            <004>
10280 S$=A$(X):A$(X)=A$(Y):A$(Y)=S$                <144>
10290 X=X+1:Y=Y-1                                  <078>
10300 GOTO 10240                                   <169>
10310 REM                                          <212>
10320 GOSUB 3000                                   <132>
10330 RG(Z+1)=Y:LG(Z+1)=LG(Z):GOSUB 10200          <212>
10340 LG(Z+1)=X:RG(Z+1)=RG(Z):GOSUB 10200          <132>
10350 Z=Z-1:RETURN                                 <205>
10360 REM                                          <006>
```

Listing 1. Der Quicksort

Sortierprogramm	Anzahl der Elemente 100	250	500
1) Straight Insertion:	45 s	290 s	1225 s
2) Bubblesort:	99 s	639 s	2771 s
3) Bubblesort 2:	99 s	661 s	3030 s
4) Straight Select:	100 s	623 s	2989 s
5) Shellsort:	37 s	108 s	295 s
6) Heapsort:	31 s	95 s	214 s
7) Quicksort:	30 s	82 s	186 s

Bild 4. Direktvergleich aller Sortierprogramme (s = Sekunden)

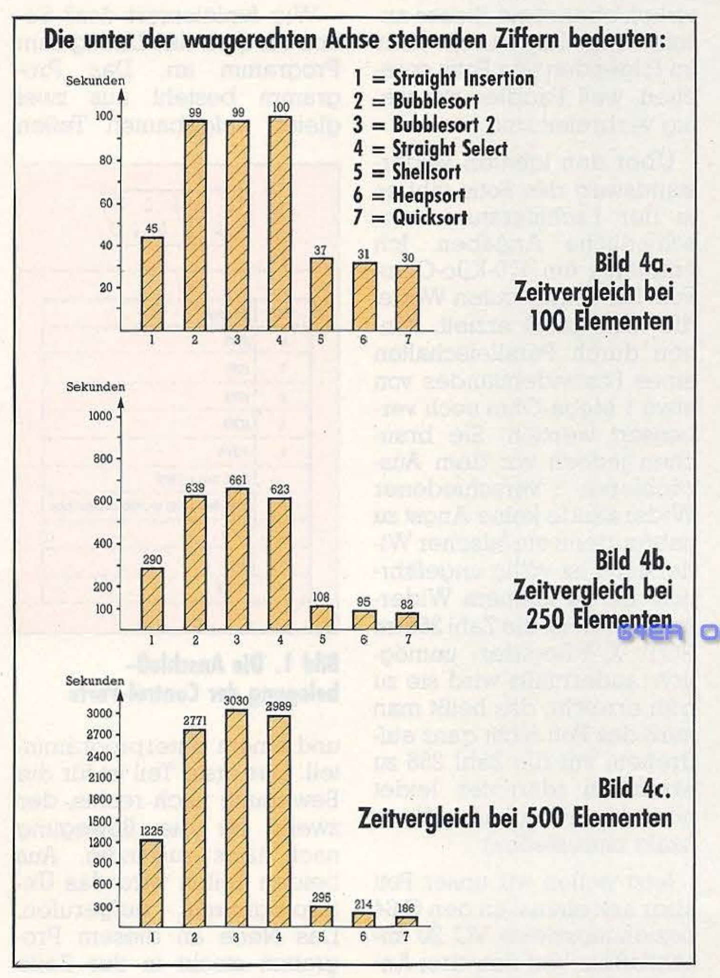


Bild 4a. Zeitvergleich bei 100 Elementen

Bild 4b. Zeitvergleich bei 250 Elementen

Bild 4c. Zeitvergleich bei 500 Elementen

```
10040 FOR X=2 TO A                              <136>
10050 IF A$(X)>=A$(X-1)THEN 10120               <038>
10070 X$=A$(X):FOR Y=X-1 TO 1 STEP-1            <117>
10080 A$(Y+1)=A$(Y)                             <110>
10090 IF X$<=A$(Y-1)THEN 10110                  <114>
10100 A$(Y)=X$:GOTO 10120                       <186>
10110 NEXT Y                                    <160>
10120 NEXT X                                    <162>
```

Listing 2. Straight Insertion

```
10040 FOR X=A-1 TO 1 STEP-1                     <063>
10050 FOR Y=1 TO X                              <006>
10060 IF A$(Y)<=A$(Y+1)THEN 10080               <003>
10070 S$=A$(Y):A$(Y)=A$(Y+1):A$(Y+1)=S$         <094>
10080 NEXT Y                                    <130>
10090 NEXT X                                    <132>
```

Listing 3. Bubblesort

```
10040 FOR X=A TO 2 STEP-1:X$=""                 <096>
10050 FOR Y=1 TO X                              <006>
10060 IF A$(Y)>X$THEN X$=A$(Y):Z=Y              <238>
10070 NEXT Y                                    <120>
10080 S$=A$(X):A$(X)=A$(Z):A$(Z)=S$             <218>
10090 NEXT X                                    <132>
```

Listing 4. Straight Select

Listing 2 bis 8. Hier sind noch einmal alle Sortieralgorithmen in optimierter Form zusammengefaßt. Beachten Sie dabei, daß das Erstellen des Sortierfeldes und die Ausgabe der Ergebnisse in dem Rahmenprogramm geschieht, das bereits in der 64'er-Ausgabe 4/85 abgedruckt wurde (Sortieren — Teil 1), aber auch auf der Leserservicediskette gespeichert ist. Beachten Sie die Eingabehinweise auf Seite 53.

```
10040 G=A-1:FOR X=A-1 TO 1 STEP-1               <148>
10050 F=0:FOR Y=1 TO G                          <116>
10060 IF A$(Y)<=A$(Y+1)THEN 10080               <003>
10070 F=Y:S$=A$(Y):A$(Y)=A$(Y+1):A$(Y+1)=S
      $                                         <143>
10080 NEXT Y                                    <130>
10090 G=F:IF F=0 THEN 50000                     <160>
10100 NEXT X                                    <142>
```

Listing 5. Bubblesort 2

```
10035 DIM AA$(A)                                <016>
10040 S=INT(A/2)                                <114>
10050 FOR X=1 TO S                              <123>
10060 FOR Y=1 TO INT(A/S)                       <074>
10070 AA$(Y)=A$((Y-1)*S+X)                      <230>
10080 NEXT Y                                    <130>
10090 AA=Y-1:GOSUB 20000                        <082>
10100 FOR Y=1 TO INT(A/S)                       <114>
10110 A$((Y-1)*S+X)=AA$(Y)                      <025>
10120 NEXT Y                                    <170>
10130 NEXT X                                    <172>
10140 S=INT(S/2)                                <090>
10160 IF S GOTO 10050                           <061>
10180 GOTO 50000                                <014>
20000 FOR XX=2 TO AA                            <239>
20010 IF AA$(XX)>=AA$(XX-1) THEN 20080          <243>
20030 XX$=AA$(XX):FOR YY=XX-1 TO 1 STEP-1       <071>
20040 AA$(YY+1)=AA$(YY)                         <137>
20050 IF XX$<=AA$(YY-1)THEN 20070               <184>
20060 AA$(YY)=XX$:GOTO 20080                    <246>
20070 NEXT YY                                   <107>
20080 NEXT XX                                   <093>
20090 RETURN                                    <080>
```

Listing 6. Shellsort

```
10040 LG=INT(A/2)+1:RG=A                        <075>
10050 IF RG<=1 THEN 50000                       <217>
10060 IF LG<=1 THEN 10110                       <195>
10080 LG=LG-1                                   <164>
10090 I=LG:GOTO 10140                           <105>
10110 S$=A$(1):A$(1)=A$(RG):A$(RG)=S$           <140>
10120 RG=RG-1                                   <165>
10130 I=1                                       <173>
10140 X$=A$(I)                                  <119>
10150 P=0                                       <205>
10160 IF 2*I<=RG AND P=0 THEN 10210             <160>
10170 A$(I)=X$                                  <028>
10180 GOTO 10050                                <047>
10210 J=2*I                                     <248>
10220 IF J<RG THEN IF A$(J)<A$(J+1)THEN J=
      J+1                                       <199>
10230 IF X$>=A$(J)THEN 10260                    <057>
10240 A$(I)=A$(J)                               <209>
10250 I=J:GOTO 10160                            <228>
10260 P=1:GOTO 10160                            <121>
```

Listing 7. Heapsort

```
10100 DIM LG(100),RG(100):Z=0:LG(1)=1:RG(1
      )=A                                       <128>
10110 GOSUB 10210                               <056>
10120 GOTO 50000                                <210>
10210 Z=Z+1:IF LG(Z)>=RG(Z)THEN 10350           <058>
10220 X=LG(Z):Y=RG(Z)                           <023>
10230 VG$=A$(INT((X+Y)/2))                      <007>
10240 IF X>Y THEN 10330                         <234>
10250 IF A$(X)<VG$THEN X=X+1:GOTO 10250         <044>
10260 IF A$(Y)>VG$THEN Y=Y-1:GOTO 10260         <150>
10270 IF X>Y THEN 10330                         <008>
10280 S$=A$(X):A$(X)=A$(Y):A$(Y)=S$             <144>
10290 X=X+1:Y=Y-1:GOTO 10240                    <116>
10330 RG(Z+1)=Y:LG(Z+1)=LG(Z):GOSUB 10210       <020>
10340 LG(Z+1)=X:RG(Z+1)=RG(Z):GOSUB 10210       <196>
10350 Z=Z-1:RETURN                              <205>
```

Listing 8. Quicksort

C 64 extern — Der Weg nach draußen (Teil 1)

Die meisten Computer-Anwender schließen nur käufliche Peripherie an ihren Homecomputer an. Oft fehlt das Wissen, wie man sich mit einfachen Mitteln neue Peripherie- und Steuergeräte selbst bauen kann. Dieser Kurs soll sowohl dem C 64- als auch dem VC 20-Besitzer einen Einstieg in diese Materie bieten.

Der C 64 und der VC 20 haben an ihren User- und Control-Ports (Joystickports) viele Anschlüsse, an denen man sehr leicht verschiedene Steuergeräte betreiben kann. Der Schwerpunkt dieses Kurses wird hauptsächlich auf dem C 64 liegen, jedoch sollen auch die VC 20-Besitzer nicht zu kurz kommen. Alle Programmbeispiele sind nämlich auf beiden Rechnern lauffähig. In dieser Folge wollen wir uns den Control-Ports zuwenden und lernen, wie man hier dem Computer Signale zuführen kann. Der Kurs wird vor allem für diejenigen interessant sein, die sich mit solchen »externen« Möglichkeiten noch nie beschäftigt haben.

Wenn Sie die Versuche experimentell nachvollziehen wollen, sollten Sie sich unbedingt je einen Stecker für den Control-Port und für den User-Port besorgen. Diese Stecker sind zwar leider recht teuer, doch für ein sicheres Zugreifen auf die einzelnen Pins des Ports unabdingbar. Sie sind, wie alle für diesen Kurs benötigten Teile, in Elektronik- oder Computerfachgeschäften erhältlich.

Wer Angst um seine Computeranlage hat, sollte vor dem Einschalten alle nicht benötigten Peripheriegeräte (Drucker, Floppy, Datasette etc.) abtrennen. Wenn Sie jedoch vorsichtig sind und genau nach den gegebenen Anweisungen vorgehen, kann nichts schiefgehen.

Als erstes wollen wir uns den beiden Anschlüssen »POT X« und »POT Y« an den Control-Ports zuwenden. Diese beiden Anschlüsse (Pin 5 und 9, siehe Bild 1) sind dazu da, dem Computer analoge Daten (analog = stufenlos, stetig; in diesem Fall Spannungswerte) mitzuteilen. Die analogen Daten werden in einem sogenannten Analog/Digital-Wandler in digitale Daten, also in eine Zahl zwischen 0 und 255 umgewandelt. Das geschieht ohne unser Zutun automatisch im Computer. Die umgewandelten Daten kann man dann aus Speicherzellen auslesen. Die Zahl in diesen Speicherzellen ist um so kleiner, je größer die Spannung an POT X/Y ist. Deshalb stehen diese Register, wenn nichts angeschlossen ist, auch immer auf 255 (= 0 Volt an POT X/Y).

Paddles im Selbstbau

Der C 64 hat an jedem seiner beiden Control-Ports je ein Anschlußpaar POT X/POT Y. Beim VC 20 gibt es nur ein solches Paar.

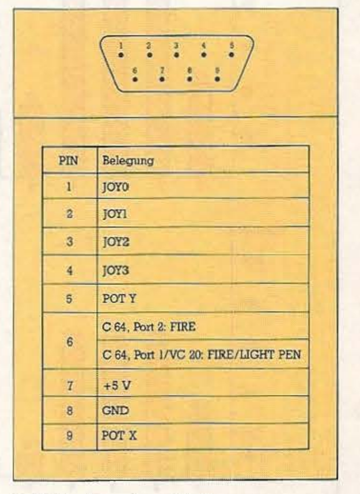
Wie sprechen wir diese Funktion des Computers an? Dazu müssen wir dem Computer also analoge Daten zuführen. Eine Möglichkeit besteht darin, über ein Potentiometer (= einstellbarer ohmscher Widerstand) verschiedene Spannungswerte an POT X oder POT Y einzustellen. Wenn Sie kein Poti (Potentiometer) zur Hand haben, können Sie in Elektronikläden eins kaufen. Falls Sie ein paar Paddles Ihr Eigen nennen, können Sie natürlich ebensogut dieses anschließen. Ich werde aber im folgenden von Potis sprechen, weil Paddles nur wenig verbreitet sind.

Über den idealen Widerstandswert des Potis gibt es in der Fachliteratur unterschiedliche Angaben. Ich empfehle ein 470-Kilo-Ohm-Poti. Die relativ guten Werte, die man damit erzielt, können durch Parallelschalten eines Festwiderstandes von etwa 1 Mega-Ohm noch verbessert werden. Sie brauchen jedoch vor dem Ausprobieren verschiedener Widerstände keine Angst zu haben, denn ein falscher Widerstand ist völlig ungefährlich: Bei zu kleinem Widerstandswert ist die Zahl 255 im POT X/Y-Register unmöglich, andernfalls wird sie zu früh erreicht, das heißt man muß das Poti nicht ganz aufdrehen, um die Zahl 255 zu erreichen (darunter leidet nur die Fähigkeit, Werte exakt einzustellen).

Jetzt wollen wir unser Poti aber erst einmal an den C 64 beziehungsweise VC 20 anschließen: Von den drei Anschlüssen des Potis verbinden Sie den mittleren mit POT X (Pin 9) und einen der verbleibenden (egal, welchen) mit +5 V (Pin 7) des Control-Ports 1 des C 64 (die VC 20-Leute natürlich mit den Anschlüssen des einzigen vorhandenen Control-Ports). Der dritte Anschluß bleibt frei. Wenn Sie Ihre Arbeit auf eventuelle Kurzschlüsse überprüft haben, schalten Sie Ihren Computer ein. Arbeiten an den Ports sollten grundsätzlich nur bei ausgeschaltetem Gerät durchgeführt werden. Tippen Sie nun das Programm »Männchendemo« (Listing 1) ab. Um das Programm an den VC 20 anzupassen, entfernen Sie eventuell vorhandene RAM-Erweiterungen und geben die Änderungen mit ein, die am Ende des Listings angegeben sind. Das gilt für alle Programme dieses Kurses.

Starten Sie nun das Programm mit »RUN«. Ein Männchen läuft über den Bildschirm. Wenn Sie am Potentiometer drehen, ändert sich die Laufgeschwindigkeit des Männchens kontinuierlich.

Wie funktioniert das? Sehen Sie sich das Listing zum Programm an. Das Programm besteht aus zwei gleich aufgebauten Teilen

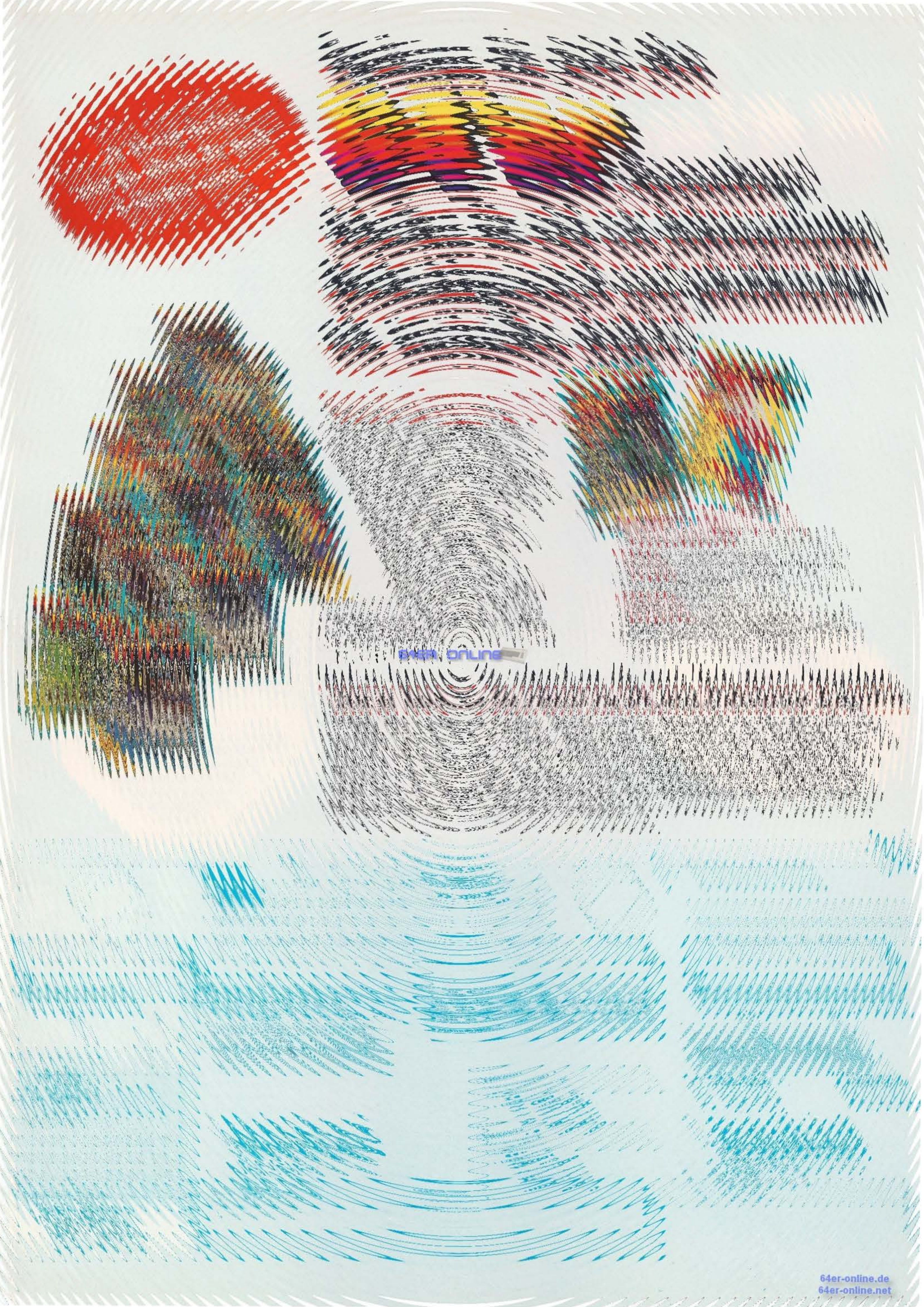


PIN	Belegung
1	JOY0
2	JOY1
3	JOY2
4	JOY3
5	POT Y
6	C 64, Port 2: FIRE C 64, Port 1/VC 20: FIRE/LIGHT PEN
7	+5 V
8	GND
9	POT X

Bild 1. Die Anschlußbelegung der Control-Ports

und einem Unterprogrammteil. Der erste Teil ist für die Bewegung nach rechts, der zweite für die Bewegung nach links zuständig. Aus beiden Teilen wird das Unterprogramm aufgerufen. Das Neue an diesem Programm steckt in der Zeile 590. Hier ist eine Warteschleife programmiert, die J von 1 bis PEEK (C) durchlaufen läßt. C steht für die Adresse, in welcher der Computer die digitale Information über den Spannungswert an POT X ablegt. Ich habe hier und auch in den folgenden Programmen bewußt viele Variablen verwendet, weil dadurch die Adaption an den VC 20 vereinfacht wird. Daher müssen zur Anpassung dieses Programms nur drei Zeilen ausgetauscht werden, weil zum Beispiel die Anfangsadresse des Bildschirm-RAMs oder die Zeilenzahl (in noch folgenden Programmen) als Variable vordefiniert wurden.

Dem POT X-Anschluß ist also beim C 64 die Adresse





```
200 REM  M A E N N C H E N D E M O         <038>
210 REM  -------------------------         <109>
220 REM  BY  TOBIAS  NICOL                 <190>
225 REM                                    <031>
230 REM ******* INITIALISIERUNG *******    <201>
240 POKE 53280,0 : POKE 53281,0            <112>
250 A = 1545 : B = 1582 : C = 54297        <060>
260 D = 55776 : E = 40                     <170>
270 PRINT "{CLR}"                          <004>
290 REM ***** FARBSPEICHER SETZEN *****    <116>
300 FOR I = D TO D+3*E                     <104>
310 POKE I,1                               <200>
320 NEXT I                                 <150>
350 REM **** BEWEGUNG NACH RECHTS  ****    <141>
360 FOR I = A TO B                         <087>
370 GOSUB 550                              <172>
380 POKE I-1,32 : POKE I+1+E,32            <110>
390 POKE I-1+E,32 : POKE I-E,32            <015>
400 NEXT I                                 <230>
420 REM ***** BEWEGUNG NACH LINKS *****    <045>
430 FOR I = B TO A STEP -1                 <000>
440 GOSUB 550                              <242>
450 POKE I+1,32 : POKE I+1+E,32            <164>
460 POKE I-1+E,32 : POKE I-E,32            <085>
470 NEXT I                                 <044>
490 REM *** RUECKSPRUNG ZUM ANFANG  ***    <060>
500 GOTO 350                               <014>
520 REM *** ENDE DES HAUPTPROGRAMMS ***    <130>
550 REM ** UNTERPROGRAMM "BEWEGUNG"  **    <071>
560 POKE I,160 : POKE I+1,77               <076>
570 POKE I-1,78 : POKE I-E,81              <122>
580 POKE I-1+E,103 : POKE I+1+E,101        <160>
590 FOR J = 1 TO PEEK (C) : NEXT J         <025>
600 RETURN                                 <150>
640 REM ******************************     <014>
650 REM *** AENDERUNGEN FUER VC-20  ***    <223>
660 REM ******************************     <034>
670 REM                                    <224>
680 REM BITTE DIESE ZEILEN ERSETZEN:       <027>
690 REM                                    <244>
700 REM 240 POKE 36879,8                   <234>
710 REM 250 A=7901 : B=7920 : C=36872      <005>
720 REM 260 D=38620 : E=22                 <150>
```

© 64'er

Listing 1. Das Männchendemo (Paddles). Alle Listings bitte mit dem neuen Checksummer aus dieser Ausgabe eingeben.

```
200 REM ** T O N D E M O **                <225>
210 REM -------------------                <232>
220 REM BY TOBIAS NICOL                    <190>
225 REM                                    <031>
230 REM ****** BILDSCHIRMAUFBAU  ******    <094>
240 PRINT "{CLR,5DOWN,7SPACE}TONDEMO"      <130>
250 PRINT "{7SPACE}-------"                <021>
260 PRINT "{3DOWN,SPACE}POT X = LAUTSTAERK
    E"                                     <226>
270 PRINT "{3DOWN,SPACE}POT Y = TONHOEHE"  <150>
290 REM ******* WERTE AUSLESEN  *******    <023>
300 POKE 56333,1                           <241>
310 POKE 56320,191                         <167>
320 L = INT (PEEK (54297) / 17)            <126>
330 T = PEEK (54298)                       <151>
340 POKE 56320,127                         <217>
350 POKE 56333,129                         <180>
370 REM ******** TON ERZEUGEN  ********    <021>
380 POKE 54296,L                           <059>
390 POKE 54279,200                         <082>
400 POKE 54280,T                           <028>
410 POKE 54285,240                         <181>
420 POKE 54283,17                          <071>
440 REM ****** WERTE DARSTELLEN  ******    <130>
450 PRINT "{HOME,19DOWN}"                  <120>
460 PRINT "{17SPACE}"                      <103>
470 PRINT "{UP,SPACE}L ="L,"T ="T          <075>
490 REM ********* RUECKSPRUNG *********    <254>
500 GOTO 290                               <070>
540 REM ******************************     <170>
550 REM *** AENDERUNGEN FUER VC-20  ***    <123>
560 REM ******************************     <190>
570 REM                                    <124>
580 REM DIE ZEILEN 300-350 UND 380-420     <168>
590 REM LOESCHEN.                          <127>
600 REM                                    <154>
610 REM NEU EINGEBEN:                      <183>
620 REM                                    <174>
630 REM 300 L=INT(PEEK(36872)/17)          <041>
640 REM 310 T=INT((PEEK(36873)*.985)/2)    <174>
650 REM     +128                           <140>
660 REM 380 POKE 36878,L                   <222>
670 REM 390 POKE 36876,T                   <127>
```

© 64'er

Listing 2. Tondemo (Paddles)

54297 und beim VC 20 die Adresse 36872 zugeordnet. Für POT Y ist bei beiden Computern jeweils die nächste Speicherzelle auszulesen, also 54298 beziehungsweise 36873. Durch die »extern« variierte Warteschleife wird die Wanderung des Männchens über den Bildschirm unterschiedlich stark verlangsamt, wodurch eine Geschwindigkeitskontrolle realisiert ist.

Probleme mit Port 2

In dem Programm »Tondemo« (Listing 2) werden dann POT X und POT Y gleichzeitig benutzt. Für den richtigen Betrieb des Programm benötigen Sie eigentlich zwei Potentiometer, Sie können jedoch auch eines abwechselnd an POT X und POT Y betreiben. Wenn Sie gerade vor einem C 64 sitzen, dann schließen Sie für dieses Programm beide Potis an den Control-Port 2 an. Ich verwende jetzt bewußt den zweiten Port, weil man hier noch ein paar Dinge beachten muß, die bei Port 1 nicht notwendig sind.

Zurück zum Programm: Mit dem X-Poti wird die Lautstärke des Tones, mit dem Y-Poti die Frequenz des Tones eingestellt. Die Werte werden in den Zeilen 320 und 330 ausgelesen und für die Tonregister aufbereitet. Bevor wir den Port 2 des C 64 auslesen können, müssen wir diesen allerdings noch dafür vorbereiten. Zunächst schalten wir die Interruptroutine des C 64 (IRQ) ab. Das geschieht durch POKE 56333,1 und ist notwendig, weil wir sonst schwankende Werte erhalten. Vielleicht ist Ihnen schon aufgefallen, daß die Register, aus denen ausgelesen wird (Zeilen 320/330), dieselben sind wie beim ersten Programm, obwohl wir doch jetzt den Port 2 beschaltet haben. Wenn Sie das bemerkt haben, können Sie sich schon denken, was jetzt noch nötig ist: Wir müssen dem C 64 mitteilen, welchen Port wir auslesen wollen. Diese Information erhält er über Bit 6 und 7 der Adresse 56320. Das verhält sich so:

	Bit 6	Bit 7
Port 1	1	0
Port 2	0	1

In dieser Adresse müssen wir also eine 127 ablegen (das ergibt sich durch die Kombination der anderen Bits in 56320), wenn wir Port 1 abfragen wollen. Dieser Wert steht hier aber sowieso immer, weshalb wir uns das Einstellen dieses Registers bei Programm 1 sparen konnten (auch das Abschalten des Interrupts ist unnötig, da sich keine Probleme ergeben). Wollen wir Port 2 auslesen, müssen wir vorher 191 nach Adresse 56320 POKEn. Erwähnenswert ist außerdem, daß dieser Wert vor jeder Abfrage neu in dieses Register eingeschrieben werden muß, da er vom Betriebssystem immer wieder mit der 127 von Port 1 überschrieben wird. Um diesen Sachverhalt nachzuprüfen, geben Sie einmal folgendes Miniprogramm ein:

10 POKE 56333,1 :
POKE 56320,191
20 PRINT PEEK (56320) :
GOTO 20

Nach RUN wird ein paarmal der Wert 191 und danach nur noch der Wert 127 ausgegeben.

Nach dem Auslesen wird dann in den Zeilen 340/350 der alte Wert wieder in das Register 56320 eingeschrieben und der Interrupt wieder eingeschaltet (POKE 56333,129).

Wenn Sie das Programm für den VC 20 anpassen, fällt Ihnen wahrscheinlich auf, daß sehr viel mehr gelöscht als neu eingefügt werden muß. Das rührt daher, daß beim C 64 sowohl die Tonerzeugung als auch das Auslesen der Port-2-Register komplizierter ist. Bei der Tonerzeugung wird das allerdings durch die weitaus bessere

Qualität der Töne wettgemacht.

Das dritte Programm, »X/Y-Steuerung« (Listing 3), zeigt eine weitere Anwendung der POT X/Y-Register. Ein Objekt (in diesem Fall ein Rechteck) kann durch die zwei Potis, die jetzt wieder an Port 1 betrieben werden müssen, an jede beliebige Position des Bildschirms gebracht werden. Die Schwierigkeit bei dieser Anwendung besteht wieder darin, die Werte, die ausgelesen werden (Zeile 270,280), geeignet aufzubereiten, also in Bildschirmkoordinaten umzurechnen. Das wird in den Zeilen 310 und 320 für die X- und Y-Richtung getan. Für den VC 20 müssen wieder nur die Variablenbelegung und die Datenaufbereitung (wegen des kleineren Bildschirms, aber der gleich großen Werte 0 bis 255) ausgetauscht werden.

Diese drei vorgestellten Programme sind für sich allein betrachtet natürlich wenig sinnvoll, um nicht zu sagen langweilig. Sie sollen jedoch auch nur eine Demonstration dafür sein, was man mit den POT X/Y-Registern machen kann und wie man sie ausliest.

```
200 REM  X / Y - S T E U E R U N G         <059>
205 REM  -------------------------          <104>
210 REM  BY  TOBIAS  NICOL                  <180>
215 REM                                     <021>
220 PRINT "{CLR}"                           <208>
230 A = 1024 : B = 55296 : C = 40           <160>
240 D = 54297                               <206>
260 REM ******* WERTE AUSLESEN  *******     <249>
270 P1 = PEEK (D)                           <178>
280 P2 = PEEK (D+1)                         <075>
300 REM ****** WERTE AUFBEREITEN ******     <090>
310 X = INT (P1/6.538)                      <056>
320 Y = INT (P2/10.625)                     <147>
340 REM ****** PUNKT VERSCHIEBEN ******     <050>
350 POKE A+XA+YA*C,32                       <125>
360 POKE A+X+Y*C,160                        <177>
370 POKE B+X+Y*C,1                          <090>
380 XA = X : YA = Y                         <100>
400 REM ********* RUECKSPRUNG *********     <164>
410 GOTO 260                                <188>
450 REM ******************************      <078>
460 REM *** AENDERUNGEN FUER VC-20  ***     <031>
470 REM ******************************      <098>
480 REM                                     <032>
490 REM FOLGENDE ZEILEN ERSETZEN:           <206>
500 REM                                     <052>
510 REM 220 PRINT "{CLR}" : POKE 36879,8    <177>
520 REM 230 A=7680 : B=38400 : C=22         <095>
530 REM 240 D=36872                         <056>
540 REM 310 X=INT(P1/12.14)                 <078>
550 REM 320 Y=INT(P2/11.59)                 <255>
```

© 64'er **Listing 3. Ein Beispiel zur X/Y-Steuerung (Paddles)**

Was man mit den Control-Ports alles machen kann ...

Eine weitere Möglichkeit in der Verwendung von POT X/Y liegt im Anschließen von »Umweltsensoren« wie zum Beispiel Licht- oder Temperaturmessern. So kann man statt des Potis einfach einen LDR (Light Dependent Resistor = lichtempfindlicher Widerstand) anschließen. Die Werte, die man aus den Registern ausliest, hängen dann von der Lichtintensität ab, die auf den LDR fällt. Man kann also durch ein Programm die Helligkeit des Raums überprüfen lassen. Bei einem Programm, das den Benutzer besonders lang vor dem Bildschirm fesselt, kann man damit einen besonderen Gag einbauen: Wenn die Sonne untergeht, und es im Zimmer dunkel wird, meldet sich der Computer mit der Information, daß man langsam das Licht einschalten sollte, um sich nicht die Augen zu verderben (das wäre doch mal etwas ganz Neues!).

Für den Lichtmesser muß man die auf den POT-Registern ausgelesenen Werte geeignet aufbereiten, wie es schon in der »Tondemo« für die Tonregister geschehen ist. Man eicht den Lichtmesser, wozu man allerdings einen bereits abgeglichenen Lichtmesser benötigt.

Um Temperaturen zu messen, nimmt man statt des lichtempfindlichen einen wärmeempfindlichen Widerstand. Diese Anwendung fällt aber genau wie der Lichtmesser schon in den Bereich der Meßtechnik. Der Phantasie sind in der Verwendung dieser Register keine Grenzen gesetzt.

Wenn Sie ein Paar Paddles besitzen, werden Sie sich vielleicht schon gefragt haben, wie man denn jetzt die beiden Feuerknöpfe an den Paddles abfragt. Diesbezüglich müssen Sie sich aber leider noch bis zur nächsten Folge dieses Kurses gedulden. Zur Zeit fehlen nämlich noch ein paar Grundkenntnisse, die zum Verstehen der Abfrage der Feuerknöpfe wichtig sind.

Zur Verwendung in Programmen eignen sich Paddles besonders dann, wenn eben zum Beispiel Geschwindigkeit oder Tonhöhe eingestellt (Männchen- und Tondemo) oder ein Objekt schnell und präzise zu einer bestimmten Stelle bewegt werden muß (X/Y-Steuerung), weil durch das Potentiometer theoretisch ein direkter Sprung zu jeder Bildschirmkoordiante möglich ist (durch schnelles Drehen), während man sich beim Joystick meistens mit einer konstanten Geschwindigkeit fortbewegt (Ausnahmen bestätigen die Regel).

Apropos Joystick: Haben Sie sich nicht schon einmal gefragt, wie Sie den Joystick in eigenen Programmen benutzen können? Ich möchte Ihnen jetzt zeigen, wie das funktioniert.

So wird der Joystick abgefragt

Während man an die POT X/Y-Anschlüsse analoge Spannungswerte anlegt, verlangen die Register, die den Joystick überwachen, klare Bedingungen: Hier gibt es nur zwei Zustände, nämlich Strom und keinen Strom beziehungsweise Eins und Null beziehungsweise +5 V (Pin 7) und GND (= Ground:negativer Pol der Computerversorgungsspannung, Pin 8). Deshalb ist der Commodore-Joystick ein sogenannter Schalter-Joystick.

Etwas anderes als der Schalter-Joystick ist der Potentiometer-Joystick, bei dem mit dem Steuerhebel Widerstandswerte auf Potentiometerbahnen eingestellt werden. Diese Potentiometer-Joysticks werden an die POT X/Y-Anschlüsse angeschlossen (es handelt sich also praktisch um zwei Paddles, die mit einem gemeinsamen »Steuerhebel« bewegt werden). Doch bleiben wir beim gewohnten Schalter-Joystick.

Durch Bewegen des Joystick-Steuerhebels können vier mögliche Kontakte geschlossen werden, davon maximal zwei gleichzeitig. Das ist bei den Zwischenrichtungen NO, SO, SW und NW der Fall. Der Feuerknopf hat einen eigenen Kontakt.

Mit diesen Kontakten stellt man die Verbindung zwischen GND (Ground, Masse) und den Control-Port-Anschlüssen 1, 2, 3, 4 oder 6 her. Welchen Richtungen diese Anschlüsse entsprechen, können Sie der Tabelle 1 entnehmen. Daß der Feuerknopf an Port 1 auf einen Anschluß namens Light Pen wirkt, lassen wir vorerst außer acht.

Im Gegensatz zu den Paddles gibt es für den Joystick im C 64 zwei Register, in denen die Joystickinformation abgelegt wird, für jeden Port also ein eigenes Register.

Der VC 20 hat zwar auch zwei Joystickregister, jedoch wird hier die Joystickinformation auf eben diese beiden Register aufgeteilt.

Bevor wir den Joystick auslesen können, müssen wir den Computer dafür vorbereiten. In der Vorgehensweise hierfür gibt es wieder erhebliche Unterschiede zwischen den C 64-Ports 1 und 2 und VC 20. Beginnen wir mit dem Port 2 des C 64: Zum Einschalten der Joystickfunktionen müssen wir die Bits 0 bis 4 der Adresse 56322 löschen. Das tun wir eleganterweise ohne die restlichen Bits zu beeinflussen mit POKE 56322, PEEK (56322) AND 224.

Zur Erinnerung: Die 224 ist die Summe der Wertigkeiten der Bits 5 bis 7 ($224 = 128 + 64 + 32 = 2^7 + 2^6 + 2^5$). Das sind genau die, die wir mit unserem POKE nicht verändern wollten. Da die restlichen Bits in 224 gleich Null

sind, werden diese im Ergebnis durch AND gelöscht, was ja unsere Absicht war!

Nachdem wir diesen POKE eingegeben haben, ist unser Computer für die Eingabe auf Joystick umgeschaltet. Sie sollten den genannten POKE übrigens niemals im Direktmodus eingeben, da danach die Tastatur Ihres Computers nicht mehr funktioniert (einschließlich RUN-STOP/RESTORE). Sie können den C 64 also nur noch ausschalten, und alles ist weg.

Beim Port 1 ist im Prinzip dieselbe Operation mit Register 56323, wie bei Port 2 mit Register 56322 nötig. Da in disem Register jedoch sowieso immer eine Null steht (also alle Bits gelöscht sind), kann man sich die Arbeit sparen.

Beim VC 20 müssen Sie
POKE 37139,0 : POKE 37154,127
eingeben, um auf Joystick umzustellen. Auch hier wird die Tastatur teilweise außer Funktion gesetzt. RUN-STOP/RESTORE bleibt Ihnen jedoch erhalten.

Was dieses ganze gePOKE im einzelnen bedeutet und warum es nötig ist, erfahren Sie in der nächsten Folge. Sie sollten jedoch noch wissen, wie Sie die Eingabe wieder auf die Tastatur legen:
C 64, Port 1: keine Eingabe nötig
C 64, Port 2: POKE 56322, PEEK (56322) OR 31
VC 20: POKE 37154,255 : POKE 37139,128

Nachdem die AND-Funktion oben schon erklärt wurde, nun noch kurz das OR: Wir wollen die Bits 0 bis 4 in 56322 setzen und die restlichen nicht verändern. Die 31 ist die Summe der Wertigkeiten der Bits 0 bis 4 (31 = 00011111 binär). Die restlichen Bits sind Null und werden daher im Ergebnis durch OR nicht verändert. Die Bits 0 bis 4 werden aber auf jeden Fall gesetzt.

Jetzt sollten wir den Joystick aber einmal abfragen, was durch die Störung der Tastatur jedoch leider im Direktmodus unmöglich ist. Die Information über den Zustand des Joystickhebels steht in folgenden Speicherstellen:
C 64, Port 1 : 56321
C 64, Port 2 : 56320

	RICHTUNG	NORDEN	SÜDEN	WESTEN	OSTEN	FEUER
VC 20 + C 64	ANSCHLUSS	JOY 0	JOY 1	JOY 2	JOY 3	FIRE
	PIN	1	2	3	4	6
C 64, PORT 1	ADRESSE	56321				
	BIT NR.	0	1	2	3	4
	WERT DES BITS	1	2	4	8	16
C 64, PORT 2	ADRESSE	56320				
	BIT NR.	0	1	2	3	4
	WERT DES BITS	1	2	4	8	16
VC 20	ADRESSE	37137			37152	37137
	BIT NR.	2	3	4	5	7
	WERT DES BITS	4	8	16	32	128

Tabelle 1. Alle wissenswerten Daten zur Joystickabfrage auf einen Blick

```
200 REM  J O Y S T I C K D E M O   1          <251>
210 REM  ----------------------------         <228>
220 REM  BY  TOBIAS  NICOL                    <190>
225 REM                                       <031>
230 REM ******* INITIALISIERUNG *******       <201>
240 X1=0 : Y1=0 : X2=0 : Y2=0 : F=0          <168>
250 POKE 53280,2 : POKE 53281,0 : D=25        <015>
260 PRINT "{CLR}" : A=1024 : B=55296 : C=4
    0                                         <151>
280 REM **** EINGABE AUF JOYSTICK  ****       <091>
290 REM **** AN PORT 1  UMSTELLEN  ****       <147>
300 POKE 56323,PEEK (56323) AND 224           <047>
320 REM **** JOYSTICKPORT AUSLESEN ****       <231>
330 J1 = PEEK (56321)                         <164>
350 REM ******* DATEN AUSWERTEN *******       <191>
360 IF (J1 AND  1) = 0 THEN Y2 = Y2-1         <049>
370 IF (J1 AND  2) = 0 THEN Y2 = Y2+1         <186>
380 IF (J1 AND  4) = 0 THEN X2 = X2-1         <055>
390 IF (J1 AND  8) = 0 THEN X2 = X2+1         <065>
400 IF (J1 AND 16) = 0 THEN F  = 1            <167>
420 REM ** KOORDINATEN KONTROLLIEREN **       <069>
430 IF X2 < 0   THEN X2 = 0                   <031>
440 IF Y2 < 0   THEN Y2 = 0                   <050>
450 IF X2 > C-1 THEN X2 = C-1                 <057>
460 IF Y2 > D-1 THEN Y2 = D-1                 <176>
480 REM ******** PUNKT BEWEGEN ********       <184>
490 IF F = 0 THEN POKE A+X1+Y1*C,32           <032>
500 POKE A+X2+Y2*C,81                         <146>
510 POKE B+X2+Y2*C,1                          <119>
530 REM ** NEUE WERTE & RUECKSPRUNG  **       <220>
540 X1 = X2 : Y1 = Y2 : F = 0                 <188>
550 GOTO 320                                  <018>
590 REM ******************************        <220>
600 REM *** AENDERUNGEN FUER VC-20  ***       <173>
610 REM ******************************        <240>
620 REM                                       <174>
630 REM AENDERN SIE DIESE ZEILEN:             <038>
640 REM                                       <194>
650 REM 250 POKE 36879,10 : D=23              <016>
660 REM 260 PRINT "{CLR}" : A=7680 : B=384
    0                                         <175>
670 REM      0 : C=22                         <006>
680 REM 300 POKE37139,0 : POKE37154,127       <146>
690 REM 330 J1 = PEEK (37137)                 <232>
700 REM 335 J2 = PEEK (37152)                 <003>
710 REM 390 IF(J2AND128)=0 THEN X2=X2+1       <025>
720 REM                                       <018>
730 REM IN DEN ZEILEN 360-380 UND 400         <134>
740 REM DIE WERTE MIT DENEN J1 "AND"-         <178>
750 REM VERGLICHEN WIRD DER REIHENFOLGE       <001>
760 REM NACH DURCH 4,8,16,32 ERSETZEN.        <167>
```


Listing 4. Demo-Programm zur Joystickabfrage

Beim VC 20 steht die Information für Osten in 37152, die für die anderen drei Richtungen und Feuer in 37137. Jedem Kontakt im Joystick ist genau ein Bit in einer dieser Speicherstellen zugeordnet. Welches Bit zu welcher Richtung gehört, zeigt Tabelle 1. Beim C 64 weisen die angegebenen Speicherstellen der beiden Ports die gleiche Bitstruktur auf. Der einzige Unterschied besteht darin, daß in Adresse 56320 das Bit 7 immer gelöscht ist.

Ein Beispiel zur Joystick-programmierung

Nach soviel Theorie wieder etwas Praxis: Geben Sie zur Übung der Joystickprogrammierung das Programm »Joystickdemo 1« (Listing 4) ein. Mit diesem Programm können Sie per Joystick einen weißen Punkt auf einem schwarzen Bildschirm in alle acht Richtungen bewegen. Ist der Feuerknopf gedrückt, wird der Punkt nicht mehr gelöscht, das heißt er zeichnet.

In Zeile 300 wird die Eingabe auf den Joystick gelegt. Zeile 330 kopiert (bei VC 20 auch Zeile 335) den Inhalt des Joystickregisters in die Variable J1 (und beim VC 20 das zweite Register in J2). Ist der Joystick in Ruhestellung, so ist das zugehörige Bit auf Eins. Bei der Bewegung in eine Richtung wird das entsprechende Bit im Joystickregister gelöscht. Bei den Zwischenrichtungen NO, SO, SW und NW werden jeweils die zwei zugehörigen Bits gelöscht. Der Feuerknopf setzt das »Fire«-Bit auf Null. Ab Zeile 350 wird J1 (und J2) ausgewertet. Wenn die bitweise Überprüfung mit AND Null ergibt, war der Joystick in dieser Richtung bewegt. Mit der Wertekontrolle ab Zeile 420 wird verhindert, daß der Punkt den Bildschirm verläßt und unkontrolliert im RAM herumschreibt. In Zeile 490 wird der alte Punkt eventuell (Abhängigkeit vom Feuerknopf) gelöscht und in 500/510 der neue gesetzt.

In diesem Programm wird der Joystick ausschließlich in Basic abgefragt. Für die hier dargestellte Anwendung ist

```
200 REM  J O Y S T I C K D E M O   2          <124>
210 REM  ---------------------------          <228>
220 REM  BY  TOBIAS  NICOL                    <190>
225 REM                                       <031>
230 REM ** MASCHINENROUTINE EINLESEN **       <132>
240 S = 0 : FOR I=842 TO 925                  <058>
250 READ A : POKE I,A : S = S+A               <144>
260 NEXT I                                    <090>
270 IF S=8788 THEN 300                        <202>
280 PRINT "{CLR,4DOWN}DATA ERROR!" : END      <172>
300 REM ******* INITIALISIERUNG *******       <017>
310 X1 = 0 : X2 = 0 : Y1 = 0 : Y2 = 0         <000>
320 POKE 53281,0 : POKE 53280,1               <069>
330 A = 1024 : B = 55296 : C = 40             <006>
340 D = 24 : PRINT "{CLR}"                    <208>
360 REM **** JOYSTICK-WERT NACH J  ****       <232>
370 SYS 842 : J = PEEK (1023)                 <244>
380 IF J = 0 OR J = 100 THEN 360              <090>
400 REM *** JOYSTICK-WERT AUSWERTEN ***       <218>
410 IF J < 100 THEN J1 = J                    <217>
420 IF J > 100 THEN J1 = J-100                <204>
430 ON J1 GOTO 450,360,460,490,470,480        <085>
440 ON J1-6 GOTO 520,500,360,360,510          <146>
450 Y2=Y2-1 : GOTO 540 : REM NORD             <218>
460 X2=X2+1 : GOTO 540 : REM OST              <047>
470 Y2=Y2+1 : GOTO 540 : REM SUED             <075>
480 X2=X2-1 : GOTO 540 : REM WEST             <077>
490 X2=X2+1:Y2=Y2-1: GOTO 540 : REM N/O       <066>
500 X2=X2+1:Y2=Y2+1: GOTO 540 : REM S/O       <209>
510 X2=X2-1:Y2=Y2+1: GOTO 540 : REM S/W       <123>
520 X2=X2-1:Y2=Y2-1: GOTO 540 : REM N/W       <002>
540 REM ***** WERTE KONTROLLIEREN *****       <001>
550 IF X2 < 0   THEN X2 = 0                   <153>
560 IF Y2 < 0   THEN Y2 = 0                   <172>
570 IF X2 > C-1 THEN X2 = C-1                 <179>
580 IF Y2 > D   THEN Y2 = D                   <038>
600 REM ******** PUNKT BEWEGEN ********       <050>
610 IF J < 100 THEN POKE A+X1+Y1*C,32         <013>
620 POKE A+X2+Y2*C,160                        <132>
630 POKE B+X2+Y2*C,1                          <241>
650 REM ** NEUE WERTE & RUECKSPRUNG  **       <084>
660 X1 = X2 : Y1 = Y2                         <211>
670 GOTO 360                                  <202>
700 REM ************ DATAS ************       <234>
710 DATA   008, 169, 000, 141, 255, 003       <020>
720 DATA   169, 001, 045, 001, 220, 208       <167>
730 DATA   008, 169, 001, 109, 255, 003       <084>
740 DATA   141, 255, 003, 169, 002, 045       <203>
750 DATA   001, 220, 208, 008, 169, 005       <238>
760 DATA   109, 255, 003, 141, 255, 003       <070>
770 DATA   169, 008, 045, 001, 220, 208       <233>
780 DATA   008, 169, 003, 109, 255, 003       <200>
790 DATA   141, 255, 003, 169, 004, 045       <063>
800 DATA   001, 220, 208, 008, 169, 006       <252>
810 DATA   109, 255, 003, 141, 255, 003       <122>
820 DATA   169, 016, 045, 001, 220, 208       <024>
830 DATA   008, 169, 100, 109, 255, 003       <162>
840 DATA   141, 255, 003, 040, 096, 000       <159>
880 REM ******************************        <000>
890 REM *** AENDERUNGEN FUER VC-20  ***       <209>
900 REM ******************************        <020>
910 REM                                       <210>
920 REM GEBEN SIE DIESE ZEILEN EIN:           <055>
930 REM                                       <230>
940 REM 240 S=0 : FOR I=842 TO 943            <116>
950 REM 270 IF S=10932 THEN 300               <231>
960 REM 320 POKE 36879,9                      <217>
970 REM 330 A=7680 : B=38400 : C=22           <043>
980 REM 340 D=22 : PRINT "{CLR}"              <133>
1000 REM BITTE GEBEN SIE STATT DEN ZEI-       <178>
1010 REM LEN 700-840 DIESE DATAS EIN:         <204>
1020 REM                                      <064>
1030 REM 700 DATA 008,169,127,141,034         <174>
1040 REM 710 DATA 145,169,000,141,019         <120>
1050 REM 720 DATA 145,141,255,003,169         <245>
1060 REM 730 DATA 004,045,017,145,208         <159>
1070 REM 740 DATA 008,169,001,109,255         <208>
1080 REM 750 DATA 003,141,255,003,169         <255>
1090 REM 760 DATA 008,045,017,145,208         <045>
1100 REM 770 DATA 008,169,005,109,255         <094>
1110 REM 780 DATA 003,141,255,003,169         <077>
1120 REM 790 DATA 128,045,032,145,208         <079>
1130 REM 800 DATA 008,169,003,109,255         <244>
1140 REM 810 DATA 003,141,255,003,169         <003>
1150 REM 820 DATA 016,045,017,145,208         <025>
1160 REM 830 DATA 008,169,006,109,255         <114>
1170 REM 840 DATA 003,141,255,003,169         <081>
1180 REM 850 DATA 032,045,017,145,208         <055>
1190 REM 860 DATA 008,169,100,109,255         <100>
1200 REM 870 DATA 003,141,255,003,169         <159>
1210 REM 880 DATA 255,141,034,145,169         <037>
1220 REM 890 DATA 128,141,019,145,040         <239>
1230 REM 900 DATA 096,000,000,000,000         <067>
```

© 64'er **Listing 5. Joystickabfrage als Maschinenroutine**

dieses System auch völlig ausreichend. Wenn es aber auf hohe Geschwindigkeit ankommt oder jede der acht Richtungen eine andere Funktion auslösen soll, der Joystick also auf jede Richtung einzeln überprüft werden muß, reicht eine reine Basic-Abfrage oft nicht mehr aus. Ich habe dieses Problem in dem Programm »Joystickdemo 2« (Listing 5) behoben. Nach dem Programmstart wird die in den DATA-Zeilen enthaltene Maschinenroutine bei beiden Computern im Kassettenpuffer abgelegt. Für den VC 20 aktivieren Sie bitte die REM-Zeilen. Da beim C 64 der Port 1 verwendet wird, sind für ihn keine Einstellungen notwendig. Die oben genannten Einstellungen für den VC 20 nimmt dessen Maschinenroutine selbsttätig vor. Daher ist sie auch etwas länger. Außerdem sind natürlich die Abfrageregister und Bitwerte angepaßt. Die Routine, die übrigens nur relative Sprünge enthält, also auch in anderen Speicherbereichen abgelegt werden kann, liest bei ihrem Aufruf die oben beschriebenen Joystickregister der Computer aus und legt den ermittelten Wert im letzten Byte des Kassettenpuffers (Adresse 1023 dezimal oder $03Ff hexadezimal) ab. Den Aufbau dieses Wertes zeigt Bild 2. Ist der Feuerknopf gedrückt, wird zu dem errechneten Wert die Zahl 100 addiert. Diese Methode hat den Vorteil, daß nach der Berücksichtigung des Feuerknopf-Wertes, also der addierten 100, (Zeilen 410/420) die Joystick-Information direkt mit ON...GOTO/GOSUB verarbeitet werden kann (Programmzeilen 430 bis 520). Da die Joystick-Werte »2«, »9« und »10« nicht vorkommen, müssen sie bei der ON...GOTO/GOSUB-Anweisung ausgeklammert

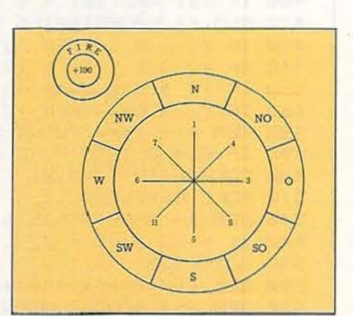


Bild 2. Diese Werte liefern die Maschinenroutinen zur Joystickabfrage (Listing 5)

werden (Sprünge nach Zeile 360 in Zeilen 430/440). Das ON...GOTO ist, nebenbei bemerkt, nur deshalb auf die Zeilen 430/440 aufgeteilt, weil eine Zeile zu voll würde. Bis auf die besprochene Maschinenroutine funktioniert das eben beschriebene Programm genauso wie die »Joystickdemo 1«.

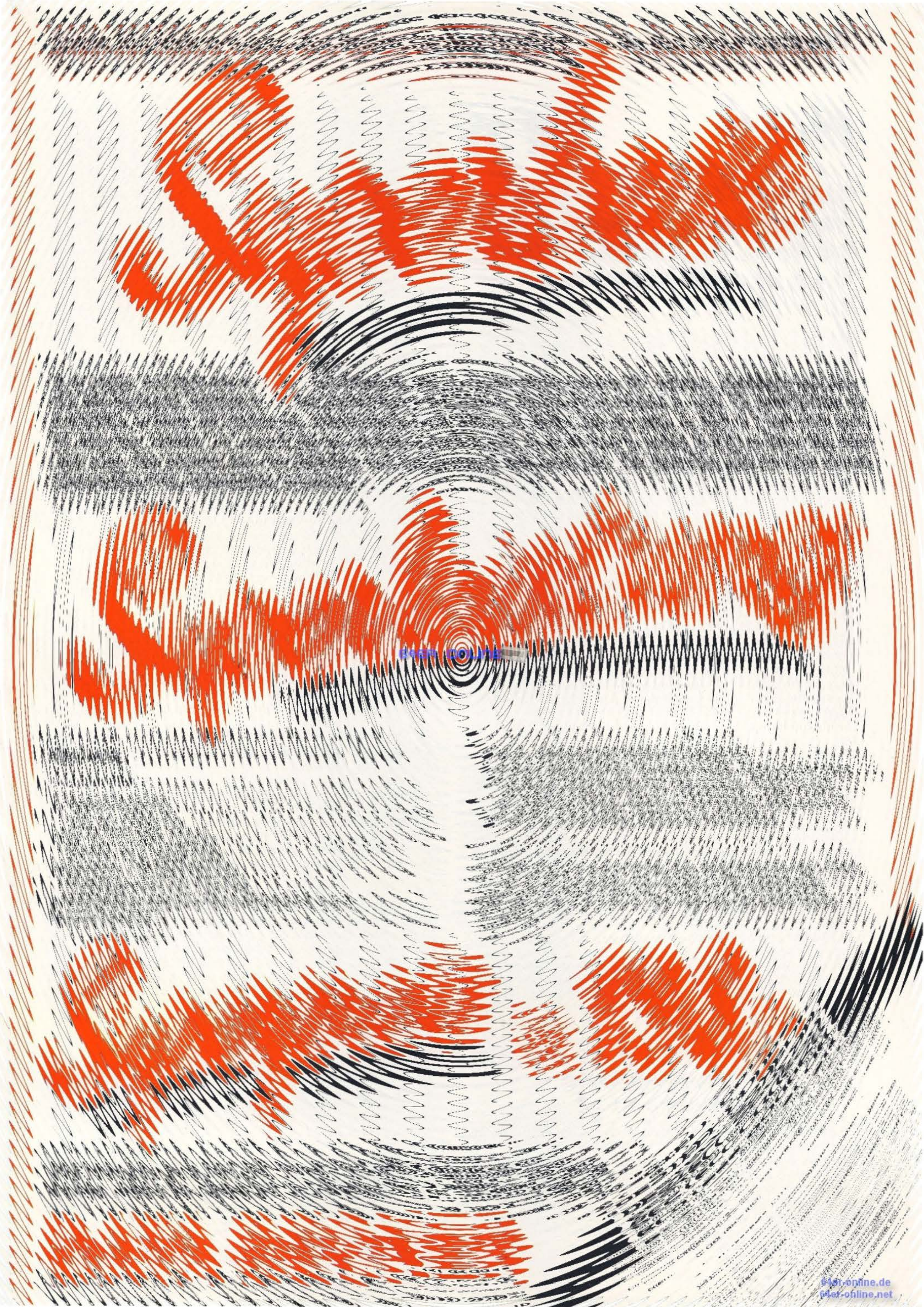
Die Geschwindigkeitsfanatiker unter Ihnen sollten jetzt noch einmal besonders gut aufpassen:

1. Obwohl die zweite Joystickdemo eine Maschinenroutine enthält, ist sie in der Ausführung etwas langsamer als die erste. Das hat einen sehr einfachen Grund: Für die hier gezeigte Anwendung ist die reine Basic-Abfrage zeitlich besser, weil bei den Zwischenrichtungen zwei Bedingungen erfüllt sind und nicht jede Richtung einzeln abgefragt werden muß.

2. Der Cursorpunkt ist schneller, wenn er die Punkte nicht löschen muß, weil dann eine Anweisung entfällt.

Das war's für heute. In der nächsten Folge werden wir die Besprechung des Control-Ports zu Ende bringen und uns anschließend dem User-Port zuwenden. Bis dahin haben Sie Gelegenheit, ausgiebig mit dem Gelernten zu experimentieren.

(Tobias Nicol/ev)



Hires-3 (Teil 3)

Text und Grafik mischen ist ein vielschichtiges Thema. Wir geben Ihnen eine ausführliche Anleitung zur Hand, mit der Sie dieses Problem relativ einfach bewältigen können.

Zwei Varianten sind denkbar, Text und Grafik zu mischen.
1.) Wir mischen auf dem Bildschirm zwei verschiedene Darstellungsmodi, nämlich den Text- und den Hochauflösungsmodus. Das ist per Rasterzeileninterrupt zu erreichen. Ein Beispiel fanden Sie in der siebten Folge des Grafikkurses (64'er, Ausgabe 10/84).
2.) Wir sorgen dafür, daß die Schrift in die Bit-Map eingetragen wird. Auch dafür sahen Sie in der siebten Folge ein einfaches Beispiel in Basic.

Wir wollen uns mal die Vor- und Nachteile der beiden Möglichkeiten vor Augen halten: Die Lösung mittels Rasterzeileninterrupt bietet den Vorzug, daß die Bildschirmaufspaltung ständig vorhanden sein kann, wenn sie einmal angeschaltet wurde. Außerdem kann man alle Textausgaben bei geeigneter Cursorsteuerung lesbar halten, sogar Fehlermeldungen oder andere unerwartete Texte. Man kann im Direktmodus trotz vorhandenem Grafikbild lesbare Eingaben machen, oder im Programm-Modus INPUT-Abfragen etc. erlauben. Als Nachteile stehen dem gegenüber: Grafik und Schrift müssen unter- oder übereinander angeordnet werden. Der Rasterzeileninterrupt ist nämlich nur zeilenweise schaltbar. Es ist also beispielsweise nicht möglich, die linke Bildschirmhälfte im Hochauflösungsmodus und die rechte im Textmodus zu verwenden. Ein weiteres Manko ist es, daß unter gewissen Umständen ein Flimmern des Bildschirms auftreten kann. Durch sorgfältiges Programmieren der Interruptroutine ist eine Quelle dafür zwar zu beseitigen, aber Probleme treten auf, wenn das Betriebssystem ein Hochscrollen des Textes erzwingt. Doch dazu später. Ein letzter Nachteil ist, daß man sehr auf andere Interruptroutinen — besonders solche, die unseren Computer funktionsfähig halten, achten muß. Aber das ist programmtechnisch lösbar.

Nun zur zweiten Variante: Da ist zunächst mal der unbestreitbare Vorzug, daß der Text an jeder beliebigen Stelle auftreten kann, ja sogar mitten in der Grafik, denn er ist ja jetzt selbst Grafik. Außerdem könnte man den Text noch vergrößern oder sonstwie anders gestalten. Das letztere — gleich hier soll's gesagt sein — ist aber in dieser Folge nicht eingeschlossen. Als Nachteile stehen dem gegenüber, daß der Text vorher definiert werden muß, daß also keine spontanen Regungen unseres Computers — wie Fehlermeldungen — damit erfaßbar sind. Außerdem sind wir es gewohnt, daß für diese Art der Test-in-Grafik-Programmierung das Zeichen-ROM in einen zugreifbaren Speicherbereich kopiert werden muß (wie zum Beispiel in Folge 7). Das aber verschlingt eine Unmenge an Speicherplatz. Weil es eines der Ziele von HIRES-3 ist, keinen unnötigen RAM-Speicher zu verschleudern, werden wir uns einer programmtechnischen Lösung dieses Problems bedienen.

Als Fazit ergibt sich, daß wir möglichst beide Versionen in Hires-3 einbauen sollten, um sowohl Fehlermeldungen und Direkteingaben als auch Text in der Grafik zu ermöglichen. Das soll Schritt für Schritt in dieser Folge geschehen. Die Programmsprache, die wir einsetzen, wird Assembler sein, und wenn Sie den Kurs zur Maschinensprache »Assembler ist keine Alchimie« lesen, dann haben Sie hier die Gelegenheit, einige Anwendungen zu erarbeiten und eventuell nach eigenen Bedürfnissen umzubauen, denn: Es gibt kein Programm, an dem nicht noch etwas zu verbessern wäre.

Rasterzeileninterrupt

Zunächst einmal, »interrupt« heißt auf deutsch »Unterbrechen«. Unser Computer — von uns unbemerkt — unterbricht viele Male pro Sekunde das, woran er gerade arbeitet, um wichtige Parameter aufzufrischen. Es gibt mehrere Sorten dieser Interrrupts, uns soll hier nur der interessieren, den wir nutzen wollen: der sogenannte IRQ. Dieser Interrupt kann softwaremäßig gestattet oder unterdrückt werden durch zwei Assembler-Befehle (SEI = setze IRQ-Flagge = Verhindern von IRQ, CLI = lösche IRQ-Flagge = Erlauben von IRQ). Außerdem kann in einigen Registern noch bestimmt werden, welche Ereignisse einen IRQ auslösen dürfen.

Sehen wir uns zuerst nochmal an, welchen Weg ein unbeeinflußter IRQ nimmt. Ganz hinten in unserem Speicher (65534/65535) steht eine Adresse (65352), die beim sogenannten Hardware-Interrupt angesprungen wird. An dieser Zieladresse 65352 werden zunächst alle Register an einen sicheren Platz gerettet, schließlich aber mittels eines indirekten Sprunges die eigentliche IRQ-Routine angesteuert. Der indirekte Sprung erfolgt zu der Adresse, die in den Speicherstellen 788/789 ($314/315) enthalten ist. Das sind RAM-Zellen, die also von uns verändert werden können. Behalten wird diese Tatsache erst mal im Gedächtnis und sehen uns den normalen weiteren Verlauf an. Normalerweise ist in diesem IRQ-Vektor als Zieladresse $EA31 (dezimal 59953) enthalten. Die an dieser Stelle startende IRQ-Routine wird im Normalfall alle 1/60 Sekunden aufgerufen. Darin wird die interne Uhr weitergestellt, der Cursor geschaltet, Ein- und Ausgabe-Parameter abgefragt, die Tastatur auf Eingaben beobachtet, etc. Abschließend holt das Programm wieder die zu Beginn geretteten Register zurück und schaltet zur normalen Tätigkeit des Computers weiter. Das Interruptprogramm ist dann bis zur nächsten 1/60 Sekunde beiseite gelegt. Diese normale Routine wird durch die Timer der CIA-Bausteine unseres Computers gesteuert.

Der übliche Weg, den auch wir beschreiten werden, ist, den Vektor 788/789 auf eine selbst programmierte IRQ-Routine zu stellen und diese dann mit einem Sprung in das Ende der normalen IRQ-Routine abzuschließen. Von dem Moment an durchläuft alle 1/60 Sekunden der Computer unsere eigene Routine.

Wie muß diese Routine aussehen? Unser Ziel soll es sein, daß eine Text-Kopfzeile auf dem Bildschirm sichtbar ist und daß die unteren vier Zeilen ebenfalls im Textmodus erscheinen. Dazwischen soll der Hochauflösungsmodus dargestellt werden (siehe Bild 1).

Das sind Aufgaben, die der VIC-II-Chip zu erledigen hat. Dafür ist er ebenfalls mit einer IRQ-Steuermöglichkeit ausgestattet. Zwei Register spielen hier die entscheidende Rolle:

53273 ($D019)	Interrupt Latch Register, auch Interrupt Request- oder Interrupt Status-Register genannt.
53274 ($D01A)	Interrupt Enable Register

Der Aufbau beider Speicherstellen ist identisch. Bit 0 ist die zum Rasterinterrupt gehörige Flagge, Bit 1 hat mit Sprite/Hintergrund-Kollisionen zu tun, Bit 2 mit Kollisionen von Sprites untereinander, Bit 3 wird bei der Lichtgriffel-Benutzung angesprochen. Die Bits 4, 5 und 6 sind unbenutzt. Bit 7 ist immer dann gesetzt (oder muß in 53274 gesetzt werden), wenn eines der anderen Bits angesprochen wird (siehe Bild 2).

Der Unterschied beider Register ist der, daß 53273 lediglich **anzeigt**, daß eine der vier möglichen Interrupt-Quellen einen IRQ ausgelöst hat. In dem Fall ist Bit 7 gesetzt, und das Bit des auslösenden Ereignisses ist gleich 1. Wir kennen sowas noch von der Folge 5, wo es um Kollisionen von Sprites ging. Bei einem Rasterzeileninterrupt findet man dann Bit 7 und Bit 0 gesetzt. Welcher Interrupt von den vier möglichen überhaupt zugelassen wird, kann man im Register 54274 bestimmen. Bit 7 regelt, ob überhaupt einer erlaubt wird (von den vier erwähnten). Ist Bit 7 gesetzt, sind solche IRQs gestattet. Durch Setzen der Bits 0 bis 3 wird die auslösende Quelle festgelegt. Dabei sind auch mehrere möglich. Man nennt das dabei gebildete Bit-Muster die Interrupt-Maske. Wenn wir nur den Rasterinterrupt zulassen wollen, müssen wir also Bit 0 und Bit 7 auf 1 setzen.

Rasterzeileninterrupt bedeutet, daß ab einer bestimmten Rasterzeile unser Computer in das Interruptprogramm springen soll, welches wir durch Einschreiben in den Vektor 788/789 definiert haben. Dazu muß dem VIC-II-Chip natürlich noch gesagt werden, welche Rasterzeile wir wählen wollen. Falls Sie über den Begriff »Rasterzeile« stolpern, dann sehen Sie in der 8. und 7. Folge der Grafik-Serie nochmal nach, wie der Computer das Bild auf Ihrem Monitor (oder Fernsehgerät) zusammenbaut. Diese Mitteilung an den VIC-II-Chip geschieht wieder über zwei Register:

53265 ($D011)	Hiervon aber nur Bit 7
53266 ($D012)	Das ganze Register

Die Sache verhält sich wie bei der X-Position von Sprites: Es können Zahlen auftreten, die größer als 255 sind. Wir haben den ganzen Bildschirm in 280 Rasterzeilen vorliegen (wobei das sichtbare Fenster etwa von Zeile 40 bis Zeile 240 reicht). Um beispielsweise die größtmögliche Rasterzeilen-Zahl 280 binär darzustellen, braucht man 9 Bits:

1 0001 1000

Dieses neunte Bit schreibt man als Bit 7 ins Register 53265, die restlichen acht Bit bilden den Inhalt des Registers 53266.

Wir planen ja die erste Zeile im Text- und den weiteren Bildschirminhalt bis zur viertletzten Zeile im Hochauflösungsmodus. Durch ein bißchen Probieren bekommt man heraus, daß der Moduswechsel einmal in der 58. Rasterzeile und dann wieder in der 218. Rasterzeile stattfinden muß. Von da an kann der Bildschirm weiter im Textmodus

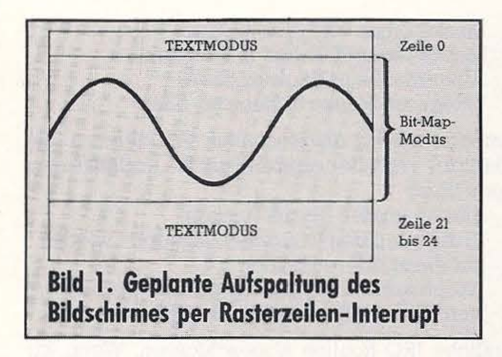
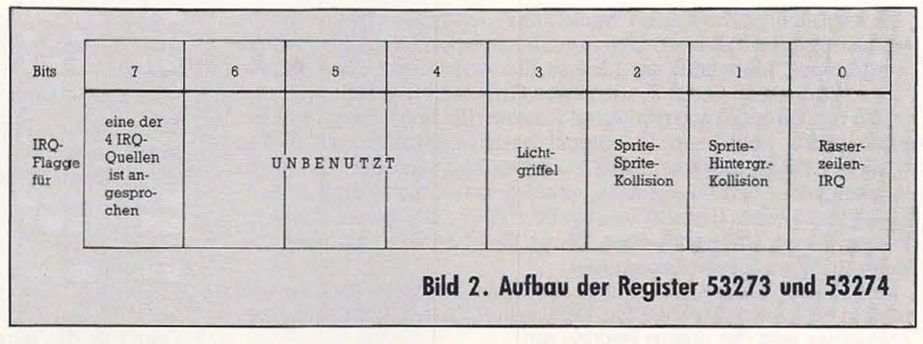


Bild 1. Geplante Aufspaltung des Bildschirmes per Rasterzeilen-Interrupt



Bits	7	6	5	4	3	2	1	0
IRQ-Flagge für	eine der 4 IRQ-Quellen ist angesprochen	UNBENUTZT			Lichtgriffel	Sprite-Sprite-Kollision	Sprite-Hintergr.-Kollision	Rasterzeilen-IRQ

Bild 2. Aufbau der Register 53273 und 53274

bleiben, bis nach dem Null-Übergang wieder Rasterzeile 58 gefunden wird. Obwohl wir letztlich den Bildschirm in drei Teile auftrennen (1. Zeile Text, dann Grafik, 21. bis 25. Zeile wieder Text) brauchen wir nur zwei Moduswechsel (Rasterzeile 58 bis 217 Grafik, Rasterzeile 218 — 57 Text). Sowohl 58 als auch 218 sind noch in acht Bit darzustellen, Bit 7 aus Register 53265 bleibt somit unverändert Null.

Jetzt wissen wir alles, was wir zur Anwendung des Rasterzeilen-Interrupt brauchen, außer... dem eigenen Interruptprogramm. Das soll nun vorgestellt werden. Zuvor aber noch eine Bemerkung an diejenigen, die (noch!) keine Assemblerprogrammierung betreiben. Die Maschinensprachroutine wird von mir ausführlich erklärt, weil man nur sehr wenig Literatur zu diesem Thema findet. Sollten Sie die Routine nutzen wollen, ohne genau wissen zu wollen, wie es programmtechnisch gemacht werden kann, dann geben Sie sie einfach nach dem beigefügten Listing (Programm 1) mittels MSE ein.

Das gesamte Programm besteht aus drei Teilen: Anschalten (Initialisieren) des Rasterzeileninterrupt, eigentliche Interrupt-Routine und Abschalten. Bei der Initialisierung muß zunächst der Inhalt der Textfenster gelöscht werden, denn dort steht ja für den Hochauflösungsmodus der Farbcode drin. Das geschieht in zwei Schleifen:

89B8 **LDA #20** Code für »Space« in Akku
89BA **LDX #27** X-Register als Index mit dezimal 39 geladen
89BC **STA 8C00,X** Leerzeichen in Zeile 0 des Bildschirmspeichers beginnt in HIRES-3 ja bei 8C00)
89BF **STA 8F48,X** und in die 21. Zeile
89C2 **DEX**
89C3 **BPL 89BC** das geschieht so lange, bis 40 Leerzeichen eingeschrieben sind, also die Zeilen 0 und 21 gelöscht wurden.

Jetzt kümmern wir uns noch um die letzten drei Zeilen:

89C5 **LDX #77** X-Indez auf 119
89C7 **STA 8F70,X** das Leerzeichen wird nun in die letzten drei Zeilen geschrieben
89CA **DEX** bis alle 120 Bildschirmpositionen gelöscht
89CB **BPL 89C7** sind.

Nun kommen wir zum Verbiegen des IRQ Zeigers:

89CD **SEI** Während dieser Prozedur können wir keine Interrupts gestatten
89CE **LDA #EC** LSB der Startadresse unserer IRQ-Routine
89D0 **STA 0314** in LSB des IRQ-Zeigers.
89D3 **LDA #89** MSB der Startadresse
89D5 **STA 0315** in MSB des IRQ-Zeigers.

Wir schreiben nun unsere erste Rasterzeile (58 = $3A) in das Rasterzeilenregister 53265/53266:

89D8 **LDA #3A** Nummer der Rasterzeile, von der an vom Text- in den Grafik-Modus umgeschaltet wird. Im weiteren obere Position genannt.
89DA **STA D012** das ist Register 53266
89DD **LAD D011** Register 53265 wird in den Akku geladen
89E0 **AND #7F** mit der AND-Maske $7F = binär 0111 1111 wird ein eventuell vorhandenes Bit 7 gelöscht
89E2 **STA D011** der so veränderte Inhalt wird ins Register zurückgeschrieben.

Als letztes in der Initialisierungsphase müssen wir noch eine Maske ins Interrupt Enable Register 53274 schreiben um anzuzeigen, daß und vor allem welchen IRQ wir zulassen:

89E5 **LAD #81** das ist binär 1000 0001
89E7 **STA D01A** das ist das IRQ Enable Register
89EA **CLI** jetzt dürfen wieder Interrupts geschehen
89EB **RTS** Rücksprung zum aufrufenden Programm.

Von nun an durchläuft jeder IRQ-Aufruf unsere ab $89EC vorhandene Routine. Das betrifft sowohl die IRQs, die von den Timern des CIA stammen, als auch die Rasterzeileninterrupts. Dann wollen wir mal schleunigst dafür sorgen, daß dort auch wirklich eine Routine steht! Zuerst überprüfen wir, ob eine Interruptanforderung auch wirklich von VIC-II-Chip her kommt:

89EC **LDA D019** Wir laden das Interrupt Request Register 53273 in den Akku
89EF **STA D019** und löschen es sofort wieder durch zurückschreiben
89F2 **BMI 89FB** war Bit 7 gesetzt, dann lag ein IRQ vom VIC-II-Chip vor, also unser Rasterzeileninterrupt. In diesem Fall überspringen wir die nächsten Zeilen.

War Bit 7 in diesem Register nicht gesetzt, dann kam die IRQ-Anforderung vom CIA und wir benutzen die normale IRQ-Routine:

89F4 **LDA DC0D** das ist das IRQ-Register des CIA und wir müssen den Anfang der normalen IRQ-Routine simulieren. Das geschieht hier durch Auslesen des CIA-Register (hier wird es auf diese Weise gelöscht)
89F7 **CLI** wir löschen die IRQ-Flagge, um während des Timer-Interrupt einen Rasterzeileninterrupt zuzulassen
89F8 **JMP EA31** wir springen zur normalen IRQ-Routine.

Nun kommt der Teil, den wir per Rasterzeileninterrupt ansteuern. Erst mal müssen wir feststellen, ob die IRQ-Anforderung durch die obere oder die untere Position erfolgt ist:

89FB **LDA D012** das ist das Rasterzeilenregister 53266
89FE **CMP #DA** $DA = dezimal 218, also die untere Position
8A00 **BCS 8A1F** kam die IRQ-Anforderung durch die untere Position zustande, dann wird zur dazugehörigen Routine verzweigt.

Nach all diesen Vorkehrungen kommt der Programmablauf hier an, wenn die obere Position für einen Rasterzeileninterrupt verantwortlich ist. Hier soll der Wechsel vom Text- zum Hochauflösungsmodus stattfinden. Für das Anschalten dieses Modus waren ja (siehe Folge 3 der Grafik-Serie) die Register 53265 ($D011) und 53272 ($D018) zuständig:

8A02 **LDA #38** Maske binär 0011 1 000 in Akku
8A04 **LDY #3B** Make binär 0011 1011 in Y-Register
8A06 **STA D018** Akku-Maske in Register 53272
8A09 **STY D011** Y-Register-Maske in Register 53265.

Damit wurde der Hochauflösungsmodus eingeschaltet und im folgenden Bildschirmteil wird der Bit-Map-Inhalt dargestellt. Es gibt nun ein Problem, das ich in den nächsten Programmzeilen einigermaßen lösen möchte. Unter dem Grafikbild werden wieder 4 Textzeilen eingerichtet. Sobald der Text dort über Zeile 24 hinausreicht, erzwingt das Betriebssystem ein Hochscrollen des Bildschirm-RAM-Inhaltes. Das drückt sich an zwei Stellen aus: Textzeilen schieben sich in den unteren Teil des Grafik-Bildes hinein, wo sie als farbige Quadrate stören. Zum zweiten scrollt der Inhalt der Kopfzeile aus dem Bildschirm und dafür treten die Farbcodes aus dem oberen Teil des Grafik-Bildes dort hinein und zeigen ein Sammelsurium von Zeichen. Für das zweite Problem, also die Zerstörung der Kopfzeile, werde ich hier keine Lösung angeben. Die finden Leser des Assembler-Kurses in der Ausgabe 2/1985 des 64'er-Magazins. Mit ein wenig Geschick können Sie das Programm zum Rückschreiben der Kopfzeile um- und hier einbauen. Aber auch das andere Problem ist zwar gelöst, aber noch nicht ganz zufriedenstellend. Wir schreiben einfach bei jedem Rasterzeilen-IRQ in den unteren Rand des Grafik-Schirmes die Farbcodes hinein:

8A0C **LDX #27** das ist wieder der Zähler, den wir schon von der Initialisierung her kennen
8A0E **LDA 8E26** aus irgendeinem Bildschirmspeicherplatz des Hochauflösungsbildes wird der Farbcode entnommen und in den Akku gelegt
8A11 **STA 8F20,X** dieser Farbcode wird in die letzte Grafikzeile geschrieben
8A14 **DEX**
8A15 **BPL 8A11** das geschieht, bis die ganze Zeile neu beschrieben wurde.

Es zeigt sich, daß auf diese Weise das Problem zwar gelöst wurde, daß sich aber bei häufigem Scrollen, zum Beispiel beim LISTen eines Programmes, manchmal ein kleines Flackern ergibt. Eine andere Möglichkeit, diese Scroll-Frage in den Griff zu bekommen, wäre natürlich die Veränderung der Scroll-Routine des Betriebssystems gewesen. Dazu hätte man allerdings die RAM-Bereiche unter den ROMs verwenden müssen, was — abgesehen von einer Unmenge verplemperten Speicherplatzes — auch Schwierigkeiten mit unserer Bit-Map unter dem Basic-ROM ergeben hätte. Falls Sie eine bessere Lösung wissen, dann schreiben Sie mir. So können wir vielleicht gemeinsam Hires-3 vervollkommnen.

Aber unser Programm ist noch nicht fertig. Wir müssen an den zweiten Moduswechsel denken. Dazu schreiben wir in das Rasterzeilenregister jetzt die untere Position ein:

8A17	**LDA #DA**	das ist die Rasterzeilen-Nummer der unteren Position
8A19	**STA D012**	da haben wir wieder unser Rasterzeilen-Register. Von nun an wird der IRQ von dieser unteren Position ausgelöst.
8A1C	**JMP EA81**	schließlich springen wir zum Ende der normalen IRQ-Routine.

Wenn jetzt der nächste Rasterzeilen-Interrupt ausgelöst wird, dann muß er auf ein Programm laufen, das den Textmodus einrichtet:

8A1F	**LDA #34**	Maske binär 0011 0100 in Akku
8A21	**LDY #1B**	Maske binär 0001 1011 in Y-Register
8A23	**STA D018**	Akku-Maske in Register 53272
8A26	**STA D011**	Y-Register-Maske in Register 53265

Damit ist der Textmodus wieder eingeschaltet. Wir müssen nun noch dafür sorgen, daß der Wert der oberen Position ins Rasterzeilenregister eingetragen wird:

8A29	**LDA #3A**	dies ist unsere obere Position
8A2B	**STA D012**	Wir stellen das Rasterzeilenregister wieder auf diese obere Position
8A2E	**JMP EA81**	Abschließend erfolgt wieder der Sprung zum Ende der normalen IRQ-Routine.

Damit ist die eigentliche IRQ-Routine abgeschlossen. Wenn wir aber jemals wieder zu normalen Verhältnissen zurückkehren wollen, dann sollten wir auch noch einen Programmteil zum Abschalten des Rasterzeileninterrupt konstruieren. Das geschieht zunächst einmal durch Löschen des Interrupt Request Registers im VIC-II-Chip:

8A31	**SEI**	Wir wollen beim Abschalten nicht durch umhervagabundierende IRQs gestört werden.
8A32	**LDA #00**	
8A34	**STA D01A**	Durch Einschreiben einer Null wird Register 53274 gelöscht

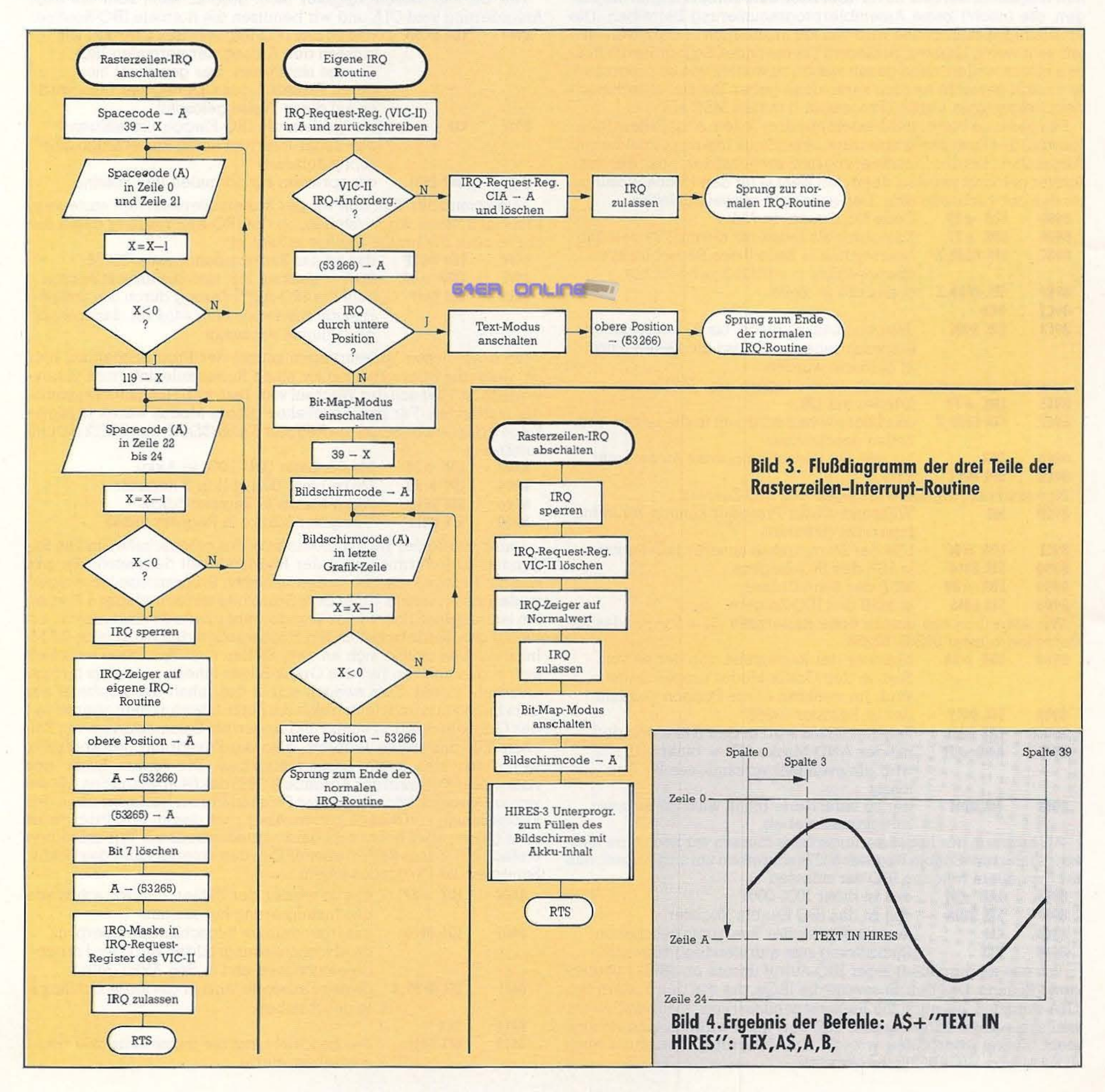


Bild 3. Flußdiagramm der drei Teile der Rasterzeilen-Interrupt-Routine

Bild 4. Ergebnis der Befehle: A$+"TEXT IN HIRES": TEX,A$,A,B,



Dann stellen wir den IRQ-Zeiger wieder auf die normale Routine $EA31:

```
8A37  LDA #31    LSB der IRQ-Adresse
8A39  STA 0314   in LSB des IRQ-Zeigers
8A3C  LDA #EA    MSB der IRQ-Adresse
8A3E  STA 0315   in MSB des IRQ-Zeigers
8A41  CLI        Jetzt darf wieder unterbrochen werden
```

Zu guter Letzt soll nach Beendigung des Rasterzeilen-Interrupt der Hochauflösungszustand wiederhergestellt werden für den ganzen Bildschirm, und die Eintragungen in den Textzeilen müssen gegen den Farbcode ausgetauscht werden:

```
8A42  LDA #38
8A44  LDY #3B
8A46  STA D018
8A49  STY D011   Das alles kennen Sie schon von weiter oben
                 im Programm (ab 8A02)
8A4C  LDA 8E26   wieder laden wir den Farbcode einer belie-
                 bigen Bildschirmspeicherzelle in den Akku
8A4F  JSR 9532   Diese Routine füllt den gesamten Bildschirm-
                 speicher mit dem Akku-Inhalt.
8A52  RTS        Rücksprung zum aufrufenden Programm
```

Das war's! Mit SYS 35256 schalten Sie die Bildschirm-Aufspaltung ein, mit SYS 35377 wieder aus. Ein Flußdiagramm dieser Routine finden Sie in Bild 3.

Als Programm 1 ist diese Routine zur Eingabe mittels MSE angefügt. Programm 3 ist ein Basic-Aufrufprogramm, das aber außer dieser neuen Implementierung noch die zweite Version beansprucht. Bevor Sie Programm 3 also starten, bauen Sie zunächst noch Programm 2, nämlich das direkte Einschreiben von Text in die Bit-Map in Hires-3 ein. Wie man alles zusammensetzt, wird Ihnen am Ende dieser Folge noch erklärt werden.

Zeichen in die Bit-Map schreiben

Zunächst legen wir fest, auf welche Weise wir die Eingaben machen wollen. Der darzustellende Text soll sowohl als Stringvariable (zum Beispiel B$), als auch als direkter String (zum Beispiel »TEST«) als auch als Stringfeldvariable (zum Beispiel B$(1)) und schließlich auch noch als Stringfunktion (zum Beispiel MID$(B$,3,2)+STR$(A)) anzugeben sein. Alle in Basic erlaubten String-Erscheinungsformen dürfen also auftreten. Weiterhin sollen Zeile und Spalte das Stringanfangs anzugeben sein. Das ganze wird schließlich noch mit einem neuen Befehlswort »TEX« in Hires-3 eingebaut (Bild 4). Doch dazu später. Die Syntax soll dann lauten: **TEX, String, Zeile, Spalte**

Die Angaben Zeile, Spalte dürfen ebenfalls in jeder erdenklichen, in Basic erlaubten Form, erscheinen. Im Programm muß dann ein Filter enthalten sein, der eine Fehlermeldung bei Falscheingaben (zum Beispiel Zeile = 234 oder ähnliches) ausgibt. Wir sind eigentlich schon mitten in der Besprechnung des ersten Teils unseres Maschinenprogrammes, nämlich der Parameterübergabe. Der zweite Teil muß nun aus den Angaben Zeile und Spalte den Ort in der Bit-Map ausrechnen, an den der String geschrieben wird. Das Startbyte in der Bit-Map ergibt sich (siehe Grafik-Folge 3) nach:

Startbyte = 320*Zeile + 8*Spalte + Anfangsadresse der Bit-Map

Nachdem das Startbyte bekannt ist, wird der String Zeichen für Zeichen durchgesehen, der ASCII-Code in den Bildschirmcode umgerechnet und schließlich in die Bit-Map eingeschrieben.

Das Umrechnen geschieht in einem kleinen Unterprogramm. Wieso eigentlich »Bildschirmcode«? Das liegt daran, daß der Bildschirmcode gleich der laufenden Nummer der Zeichen im Zeichen-ROM ist. Wie diese Zeichen dort aussehen, hatten wir uns schon in der 2. Folge der Grafik-Serie angesehen.

Auch das Einschreiben in die Bit-Map geschieht in einem Unterprogramm. Wo holen wir die Zeichen hier, wenn wir nicht bereit sind, das Zeichen-ROM ins RAM zu kopieren? Aus dem Zeichen-ROM selbst. Um das direkt lesen zu können, muß jeweils der Prozessorport ($01) so geschaltet werden, daß das Zeichen-ROM zugreifbar wird. Dr. H. Hauck hat's in seiner »Memory Map mit Wandervorschlägen«, 64'er, Ausgabe 11(1984) Seite 136 gut erklärt: Man erreicht das durch Löschen des Bit 2 im Prozessorport. Doch nun genug der Überlegungen, schreiben wir unser Programm!

Aus programmtechnischen Gründen taucht hier zuerst die Fehlerbehandlung auf:

```
8A54  LDX #0E    Fehlercode für ILLEGAL QUANTITY
8A56  JMP A437   Interpreter-Routine für die Ausgabe einer
                 Fehlermeldung, deren Code im X-Register
                 enthalten ist.
```

Hier fängt nun unser eigentliches Programm an mit der Übernahme der Parameter. Zunächst erfassen wir den String:

```
8A59  JSR AEFD   Interpreter-Routine, die auf Komma prüft.
8A5C  JSR AD9E   Interpreter-Routine, die einen Ausdruck aus-
                 wertet. Wenn der Ausdruck ein String ist,
                 wird in $64/65 ein Zeiger auf den String-
                 deskriptor eingerichtet.
```

64/65 ist eine häufig benutzte Speicheradresse. Wir lesen daher den Stringdeskriptor und lagern die Stringlänge in $24, die Stringstartadresse nach $04/05:

```
8A5F  LDY #00     Zähler auf Null
8A61  LDA (64),Y  Stringlänge in Akku
8A63  STA 24      und nach $24
8A65  INY         Zähler erhöhen
8A66  LDA (64),Y  LSB des Stringzeigers in Akku
8A68  STA 04      und nach $04
8A6A  INY         Zähler erhöhen
8A6B  LDA (64),Y  MSB des Stringzeigers in Akku
8A6D  STA 05      und nach $05
```

Damit ist der String gesichert, nun lesen wir die Zeilennummer:

```
8A6F  JSR AEFD   kennen wir schon: Auf Komma prüfen.
8A72  JSR AD9E   kennen wir ebenfalls, kann aber noch mehr
                 als nur Strings auszuwerten. Hier erkennt
                 diese Routine, daß eine Zahl vorliegt und
                 packt diese in den FAC (Fließkomma-
                 Akkumulator 1)
8A75  JSR B1AA   Interpreter-Routine: Wandelt den FAC-Inhalt
                 in eine 2-Byte-Integer-Zahl um. MSB landet
                 im Akku, LSB im Y-Register. Das MSB brau-
                 chen wir nicht.
8A78  CPY #19    Ist Zeile größer oder gleich dezimal 25?
8A7A  BCS 8A54   Wenn ja, Sprung zur Fehlermeldungsaus-
                 gabe
```

Zugegeben, wenn das Programm hier gelandet ist, können sich immer noch einige Falscheingaben durchgeschmuggelt haben. Aber wer wird bei der Zeilennummer-Eingabe zum Beispiel eine negative Zahl wählen! Wenn Sie Lust haben, dann können Sie ja auch noch andere Fehlereingabe-Filter einbauen. Uns soll's so erst mal genügen. Weil wir die Zahl jetzt gerade in so schön greifbarer Form haben, berechnen wir auch gleich noch den Teil »320*Zeile« für die Position in der Bit-Map. Für diese Multiplikation verwenden wir eine Hires-3-Routine:

```
8A7C  STY 5B     Zeile nach $5B
8A7E  LDA #40    LSB der Zahl 320
8A80  STA 59     nach $59
8A82  LDA #01    MSB der Zahl 320
8A84  STA 5A     nach $5A
8A86  JSR 9410   Hires-3-Routine, die eine in $59/5A liegende
                 Zahl mit einer in $5B liegenden multipliziert.
                 Das Ergebnis findet man in $57/58.
```

320*Zeile ist nun in $57/58 gespeichert und wir übernehmen die Spaltenangabe ins Programm:

```
8A89  JSR AEFD   Wie gehabt: Komma prüfen
8A8C  JSR AD9E   Kennen wir auch schon
8A8F  JSR B1AA   Bekannt: Bringt Spalte ins Y-Register
8A92  CPY #28    Ist die Zahl größer oder gleich dezimal 40?
8A94  BCS 8A54   Wenn ja, dann Sprung zur Fehlermeldungs-
                 ausgabe.
```

Hier gilt dasselbe, was für Zeile oben gesagt wurde. Und auch hier rechnen wir gleich den Ausdruck »8*Spalte« aus:

```
8A96  TYA        Spalte in Akku
8A97  CLC        Von hier an erfolgt die
8A98  ROL        Multiplikation mit 8
8A99  ROL        und das Ergebnis landet
8A9A  ROL        in den Speicherstellen
8A9B  STA 25     $25 (LSB)
8A9D  LDA #00    und
8A9F  ROL        $26 (MSB)
8AA0  STA 26
```

Nun addieren wir die beiden Ausdrücke (320*Zeile + 8*Spalte):

```
8AA2  CLC
8AA3  LDA 57     LSB von 320*Zeile
8AA5  ADC 25     +LSB von 8*Spalte
8AA7  STA 25     nach $25
8AA9  LDA 58     MSB von 320*Zeile
8AAB  ADC 26     +MSB von 8*Spalte + Carry
8AAD  STA 26     nach $26
```

Schließlich zählen wir noch die Bit-Map-Startadresse dazu:

```
8AAF  CLC
8AB0  LDA 25     LSB von 320*Zeile+8*Spalte
8AB2  ADC #00    +LSB Bit-Map-Start
8AB4  STA 25     Ergebins=LSB Stringstart in der Bit-Map
                 nach $25
8AB6  LDA 26     MSB von 320*Zeile+8*Spalte
8AB8  ADC #A0    +MSB Bit-Map-Start
8ABA  STA 26     MSB des Stringstarts in der Bit-Map nach $26
```

Damit haben wir sowohl die Parameterübergabe als auch die Positionierung in der Bit-Map programmiert:
Wir finden nun in
$24 die Stringlänge,
$04/05 die Startadresse des Strings im Speicher
$25/26 die Startadresse des Strings in der Bit-Map

Wir kommen nun zu dem Programmteil, in dem der String Zeichen für Zeichen gelesen, umgerechnet und schließlich gedruckt wird:

8ABC	LDY #00	Zähler auf Null
8ABE	LDA (04),Y	String-Zeichen in den Akku lesen,
8AC0	TAX	und ins X-Register schieben
8AC1	TYA	Zähler in den Akku
8AC2	PHA	und auf den Stapel retten
8AC3	TXA	Akku-Inhalt wiederherstellen
8AC4	JSR 8AD2	Unterprogramm, das die Umrechnung des ASCII-Codes im Akku zum Bildschirmcode vornimmt
8AC7	JSR 8AF6	Unterprogramm, welches das Übertragen der Zeichen aus dem Zeichen-ROM in die Bit-Map durchführt
8ACA	PLA	Zähler vom Stapel holen
8ACB	TAY	und wieder ins Y-Register schreiben
8ACC	INY	Zähler erhöhen
8ACD	CPY 24	Vergleich des Zählers mit der Stringlänge
8ACF	BMI 8ABE	Stringlänge erreicht? Wenn ja, dann...
8AD1	RTS	Programmende und zurück ins aufrufende Programm.

Nun kommen wir noch zu den beiden Unterprogrammen. Zunächst die Umrechnung vom ADCII- in den Bildschirmcode. Der Code des eingelesenen Zeichens befindet sich im Akku:

8AD2	BPL 8AD7	Liegt ein geSHIFTetes Zeichen vor? Dann ist nämlich Bit 7 gesetzt. Wenn Bit 7 nicht gesetzt ist, erfolgt der Sprung
8AD4	JMP 8AE6	ansonsten wird hier zur Routine für SHIFT-Zeichen gesprungen

Jetzt haben wir's also mit nicht geSHIFTeten Zeichen zu tun:

8AD7	CMP #20	haben wir es etwa mit Steuerzeichen zu tun?
8AD9	BCC 8AF3	Wenn ja, dann verzweigen wir
8ADB	CMP #60	liegen Grafikzeichen vor?
8ADD	BCC 8AE3	wenn nein, dann Sprung
8ADF	AND #DF	mit der Makse 1101 1111 wird Bit 5 gelöscht
8AE1	BNE 8AE5	unbedingter Sprung
8AE3	AND #3F	mit der Maske 0011 1111 werden die Bits 6 und 7 gelöscht
8AE5	RTS	Fertig mit den ungeSHIFTeten Zeichen. Rücksprung ins aufrufende Programm.

Im folgenden bearbeiten wir die SHIFT-Zeichen:

8AE6	AND #7F	Mit der Maske 0111 1111 wird Bit 7 gelöscht
8AE8	CMP #7F	liegt das Pi-Zeichen vor?
8AEA	BNE 8AEE	Wenn nicht, Sprung
8AEC	LDA #5E	wenn ja, dann Code für das Pi-Zeichen in den Akku
8AEE	CMP #20	liegt ein Steuerzeichen vor?
8AF0	BCC 8AF3	Wenn ja, Sprung
8AF2	RTS	wenn nein, dann ist jetzt der Bildschirmcode im Akku und wir springen zurück zum aufrufenden Programm.

Nun haben wir es nur noch mit den Steuerzeichen zu tun. Die ignorieren wir und setzen dafür einfach ein Leerzeichen ein:

8AF3	LDA #20	Code für »Space« in Akku
8AF5	RTS	und Rücksprung zum aufrufenden Programm.

Kommen wir nun zum zweiten Unterprogramm, das die Zeichen, welche im Akku enthalten sind als Bildschirmcodes, aus dem Zeichen-ROM heraus und in die richtige Stelle der Bit-Map hineinliest:

8AF6	LDX #00	
8AF8	STX 27	LSB der Zeichen-ROM-Startadresse = 0
8AFA	STX 29	Zwischenspeicher auf Null
8AFC	LDX #D0	MSB-Zeichen-ROM-Startadresse
8AFE	STX 28	nach $28
8B00	CLC	Von hier an wird der
8B01	ROL	Zeichencode im Akku
8B02	ROL 29	mal 8 gerechnet
8B04	ROL	Zu guter Letzt findet
8B05	ROL 29	man das LSB des
8B07	ROL	Ergebnisses im Akku
8B08	ROL 29	und das MSB in $29
8B0A	CLC	Von hier an wird die
8B0B	ADC 27	Startadresse des
8B0D	STA 27	Zeichenmusters im ROM
8B0F	LDA 28	berechnet und in
8B11	ADC 29	$27/28 abgelegt
8B13	STA 28	

Damit wissen wir nun, daß 8 Bytes von der Adresse $27/28 im Zeichen-ROM zur Adresse $25/26 in der Bit-Map übertragen werden müssen. Das geschieht nun:

8B15	LDY #00	Y-Index auf Null
8B17	LDX #08	X-Register-Zähler auf 8
8B19	LDA 01	Prozessorport-Inhalt in Akku
8B1B	PHA	und auf den Stapel beiseitelegen
8B1C	AND #FB	mit Maske binär 1111 1011 Bit 2 löschen = Zeichen-ROM zugreifbar machen
8B1E	SEI	wir können jetzt keine Interrupts gebrauchen
8B1F	STA 01	den neuen Prozessorport-Inhalt einlesen
8B21	LDA (27),Y	das Zeichen-Muster Byte für Byte aus dem Zeichen-ROM herauslesen in Akku
8B23	STA (25),Y	und in Bit-Map einschreiben
8B25	INY	Y-Index erhöhen
8B26	DEX	X-Zähler vermindern
8B27	BNE 8B21	wiederholen, bis X-Zähler gleich Null
8B29	PLA	alten Prozessorport Inhalt vom Stapel zurückholen
8B2A	STA 01	und rekonstruieren
8B2C	CLI	jetzt darf wieder unterbrochen werden
8B2D	CLC	Ab hier wird die
8B2E	LDA 25	Zieladresse in der
8B30	ADC #08	Bit-Map um 8 erhöht
8B32	STA 25	$25/26 enthält dann
8B34	LDA 26	schon für das nächste
8B36	ADC #00	einzuschreibende
8B38	STA 26	Zeichen die aktuelle Adresse.
8B3A	RTS	Ende des Unterprogrammes. Rücksprung ins aufrufende Programm.

Damit hätten wir's. Als Programm 2 finden Sie — falls Sie ohne Assembler (zum Beispiel SMON) arbeiten — ein mittels MSE eintippbares Listing dieser Routine und für den Überblick ist als Bild 5 noch ein komplettes Flußdiagramm gezeigt.

Wir setzen das Puzzle zusammen

Um nun diese beiden Programmteile in Hires-3 einzubinden, sollten zunächst Programm 1 und Programm 2 abgetippt und gespeichert werden. Anschließend laden Sie Hires-3 (mit Load"HIRES-3", 8,1 beziehungsweise ,1,1 bei Kassettenbetrieb), geben die Schutz-POKEs ein:
POKE52,128:POKE56,128 und anschließend NEW. Das wurde in der Folge 8 der Grafikserie versehentlich ausgelassen. Nun laden Sie ebenfalls absolut (also mit LOAD"PROGRAMM 1",8,1 oder ,1,1) das Programm 1 ein, geben wieder NEW ein, laden dann absolut (!) das Programm 2 ein und schließen das alles mit einem letzten NEW ab. Hires-3 und die beiden Ergänzungen stehen nun nahtlos aneinandergefügt im Speicher. Um den TEX-Befehl zu ermöglichen, muß nun noch mittels einiger POKEs Hires-3 etwas verändert werden. Geben Sie also bitte noch ein:
POKE37694,89:POKE37695,138
POKE37858,84:POKE37859,69:POKE37860,88:POKE37861,0:POKE37862,0

Mit Hilfe des SMON oder eines anderen dazu fähigen Monitors können Sie das so ergänzte Programm Hires-3 nun komplett abspeichern, zum Beispiel beim SMON mit dem Kommando: S"HIRES-3", 08,8000,9DCB

Es wird Zeit, Hires-4 zu entwickeln. In der Befehlsliste von Hires-3 ist nämlich kein Byte mehr Platz gebleiben, um alle Optionen, die nun mit SYS-Kommandos aufgerufen werden, durch neue Befehlsworte anzusprechen. TEX war das letzte neue Wort, das gerade noch hineinpaßte. So rufen Sie nun alles Neue auf:

SYS 35256 Bildschirmaufspaltung durch Rasterzeileninterrupt anschalten.

SYS 35377 Bildschirmaufspaltung wieder ausschalten. Diesen SYS-Befehl müssen Sie sich gut merken. Wenn mitten im Programm der Computer bei aufgespaltenem Bild durch einen Fehler aussteigt, können Sie nämlich durch dieses SYS 35377 und anschließendes HOF wieder in den normalen Modus gelangen.

TEX,String,Zeile,Spalte Einschreiben eines durch String definierten Textes an die Stelle Zeile, Spalte in die Bit-Map. So setzt der Befehl TEX,"HALLO",10,12 den Text HALLO in die 10. Zeile ab Spalte 12.

Die Bildschirmaufspaltung sollte nicht zusammen mit dem UHR-Befehl und der Hardcopy-Routine betrieben werden. Hires-3 ist nämlich noch nicht darauf eingerichtet, mehrere Interrupt-Routinen parallel zu verarbeiten. Als Programm 3 finden Sie noch ein Basic-Demonstrationsprogramm, das einige Anwendungen der neuen Befehle erläutert.

(Heimo Ponnath/gk)



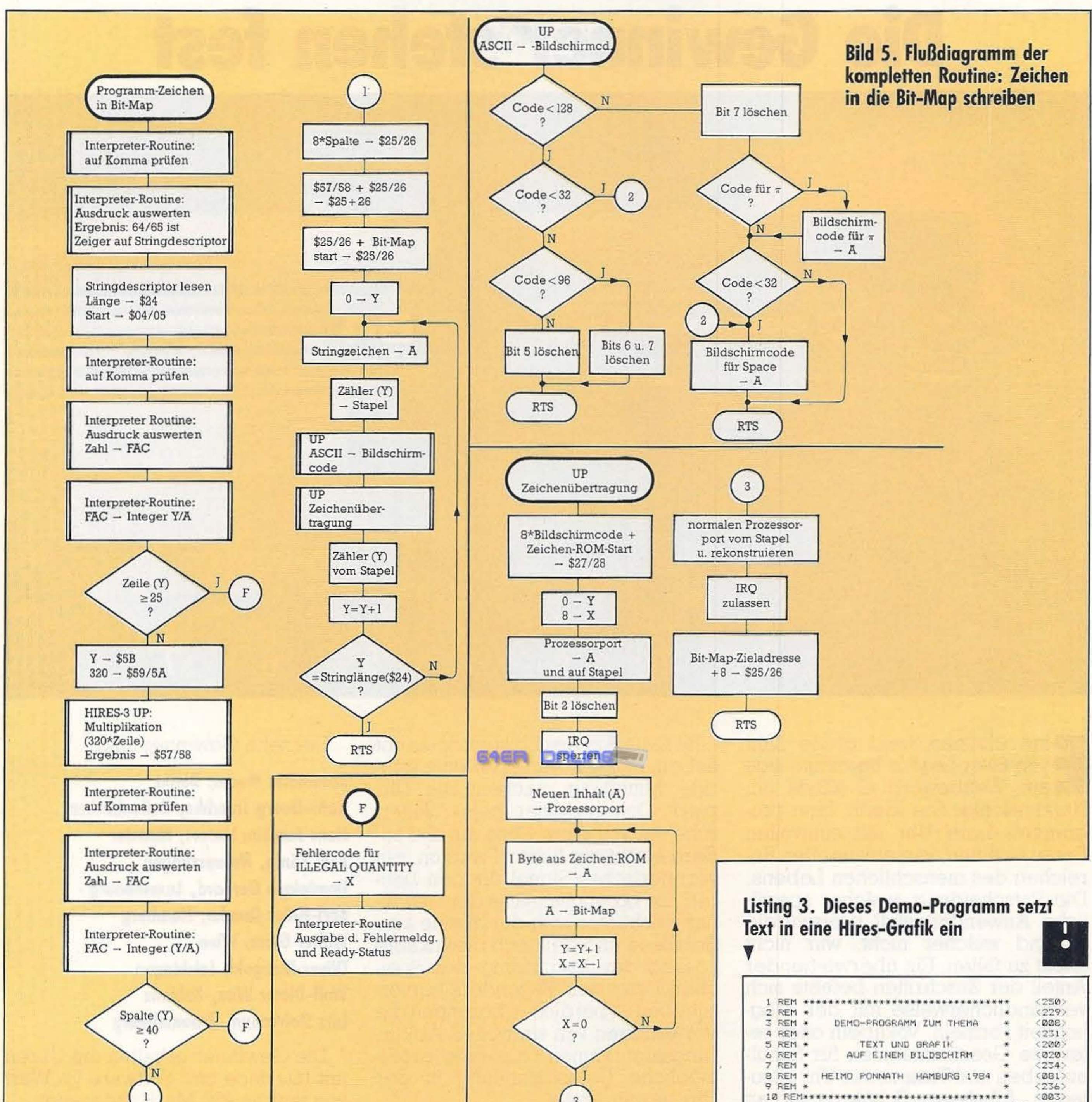


Bild 5. Flußdiagramm der kompletten Routine: Zeichen in die Bit-Map schreiben

Listing 1. Die Interrupt-Routine. Beachten Sie die Eingabe-Hinweise auf Seite 53/54 ▼

```
programm : programm1          89b8 8a53
----------------------------------------
89b8 : a9 20 a2 27 9d 00 8c 9d  46
89c0 : 48 8f ca 10 f7 a2 77 9d  32
89c8 : 70 8f ca 10 fa 78 a9 ec  a8
89d0 : 8d 14 03 a9 89 8d 15 03  bd
89d8 : a9 3a 8d 12 d0 ad 11 d0  a4
89e0 : 29 7f 8d 11 d0 a9 81 8d  ca
89e8 : 1a d0 58 60 ad 19 d0 8d  8e
89f0 : 19 d0 30 07 ad 0d dc 58  c6
89f8 : 4c 31 ea ad 12 d0 c9 da  d2
8a00 : b0 1d a9 38 a0 3b 8d 18  fa
8a08 : d0 8c 11 d0 a2 27 ad 26  e3
8a10 : 8e 9d 20 8f ca 10 fa a9  d3
8a18 : da 8d 12 d0 4c 81 ea a9  27
8a20 : 34 a0 1b 8d 18 d0 8c 11  79
8a28 : d0 a9 3a 8d 12 d0 4c 81  e9
8a30 : ea 78 a9 00 8d 1a d0 a9  01
8a38 : 31 8d 14 03 a9 ea 8d 15  e8
8a40 : 03 58 a9 38 a0 3b 8d 18  2b
8a48 : d0 8c 11 d0 ad 26 8e 20  43
8a50 : 32 95 60 ff 00 ff 00 ff  65
```

Listing 2. Direktes Einschreiben von Text in die Bit-Map ►

```
programm : programm2          8a54 8b3b
----------------------------------------
8a54 : a2 0e 4c 37 a4 20 fd ae  98
8a5c : 20 9e ad a0 00 b1 64 85  75
8a64 : 24 c8 b1 64 85 04 c8 b1  e4
8a6c : 64 85 05 20 fd ae 20 9e  eb
8a74 : ad 20 aa b1 c0 19 b0 d8  5b
8a7c : 84 5b a9 40 85 59 a9 01  ec
8a84 : 85 5a 20 10 94 20 fd ae  e0
8a8c : 20 9e ad 20 aa b1 c0 28  f6
8a94 : b0 be 98 18 2a 2a 2a 85  74
8a9c : 25 a9 00 2a 85 26 18 a5  10
8aa4 : 57 65 25 85 25 a5 58 65  54
8aac : 26 85 26 18 a5 25 69 00  4b
8ab4 : 85 25 a5 26 69 a0 85 26  f8
8abc : a0 00 b1 04 aa 98 48 8a  ef
8ac4 : 20 d2 8a 20 f6 8a 68 a8  ab
8acc : c8 c4 24 30 ed 60 10 03  2e
8ad4 : 4c e6 8a c9 20 90 18 c9  ea
8adc : 60 90 04 29 df d0 02 29  89
8ae4 : 3f 60 29 7f c9 7f d0 02  6d
8aec : a9 5e c9 20 90 01 60 a9  21
8af4 : 20 60 a2 00 86 27 86 29  fb
8afc : a2 d0 86 28 18 2a 26 29  6b
8b04 : 2a 26 29 2a 26 29 18 65  a8
8b0c : 27 85 27 a5 28 65 29 85  d2
8b14 : 28 a0 00 a2 08 a5 01 48  23
8b1c : 29 fb 78 85 01 b1 27 91  6f
8b24 : 25 c8 ca d0 f8 68 85 01  65
8b2c : 58 18 a5 25 69 08 85 25  d6
8b34 : a5 26 69 00 85 26 60 ff  52
```

Listing 3. Dieses Demo-Programm setzt Text in eine Hires-Grafik ein ▼

```
1 REM **********************************    <250>
2 REM *                                *    <229>
3 REM *    DEMO-PROGRAMM ZUM THEMA     *    <008>
4 REM *                                *    <231>
5 REM *        TEXT UND GRAFIK         *    <200>
6 REM *      AUF EINEM BILDSCHIRM      *    <020>
7 REM *                                *    <234>
8 REM *  HEIMO PONNATH  HAMBURG 1984   *    <081>
9 REM *                                *    <236>
10 REM**********************************    <003>
15 CLR:PRINT CHR$(147):Z=10:S=10:GOSUB 100
   0:PRINT"ZUVOR NOCH EINE FRAGE:":PRINT   <121>
20 INPUT"IST HIRES-3 KOMPLETT GELADEN (J/N
   )";A$                                   <192>
25 IF A$="N"THEN PRINT:PRINT CHR$(18)"BRAU
   CHEN SIE ABER!"CHR$(146):END            <246>
30 POKE 52,128:POKE 56,128:SYS 37498:PRINT
    CHR$(147)                              <084>
40 REM ++++++ SINUSKURVE ZEICHNEN +++++    <003>
45 DEF FN A(X)=SIN(X):XU=-2*π:XO=2*π:YU=-2
   .5:YO=2:TRS,XU,XO,YU,YO:HFL,6,14        <049>
50 FUNKT,A,XU,XO:TLN,XU,0,XO,0:TLN,0,YU,0,
   YO:REC,0,0,319,199                      <081>
55 REM ++++++ DER TEX-BEFEHL ++++++++++    <021>
60 TEX,"DIES IST EINE SINUSKURVE",3,8      <181>
65 REM +++++ BILDSCHIRMAUFSPALTUNG ++++    <126>
70 SYS 35256:SYS 34647:Z=21:S=0:GOSUB 1000
   :PRINT"WUENSCHEN SIE EINE SKALIERUNG";  <098>
75 INPUT"(J/N)";A$:IF A$="N"THEN 115       <120>
80 REM +++++++ SKALIERUNG  ++++++++++++    <005>
85 DEF FN X(X)=INT(39*(X+2*π)/(4*π)):DEF F
   N Y(Y)=INT(24*(2-Y)/4.5)                <123>
90 FOR X=-6 TO 6:TLN,X,0,X,-.1:X$=STR$(X):
   A=FN Y(-.3):B=FN X(X)                   <147>
95 TEX,X$,A,B:NEXT X                       <231>
100 TLN,0,1,-.2,1:TLN,0,-1,-.2,-1:A=FN Y(1
    ):B=FN X(.5):TEX,"1",A,B:A=FN Y(-1)    <245>
105 B=FN X(.3):TEX,"-1",A,B:CLR:INPUT"HARD
    COPY (J/N)";A$:IF A$="J"THEN GOSUB 200
    0                                      <043>
110 REM +++++ PROGRAMMENDE ++++++++++++    <089>
115 Z=21:S=0:GOSUB 1000:PRINT"GEBEN SIE NA
    CH READY EIN:SYS35377:HOF "            <246>
120 END                                    <248>
999 REM ++++++ UP CURSOR SETZEN +++++++    <004>
1000 POKE 214,Z:POKE 211,S:SYS 58640:RETUR
     N                                     <042>
1999 REM +++++++ UP HARDCOPY ++++++++++    <063>
2000 SYS 35377:OPEN 1,4,10:PRINT#1:CLOSE 1
     :SYS 34865:SYS 35256:RETURN           <010>
```

Die Gewinner stehen fest



Eine überraschend große Zahl von 64'er-Lesern beteiligte sich am Wettbewerb »2 KByte am Handgelenk«. Die Ideen, eine programmierbare Uhr mit sinnvollen Daten zu füllen, kamen aus allen Bereichen des menschlichen Lebens. Die Entscheidung, welcher spezifische Anwendungsfall preiswürdig ist und welcher nicht, war nicht leicht zu fällen. Ein überwiegender Anteil der Zuschriften befaßte sich verständlicherweise mit der Möglichkeit Formeln, Vokabeln oder relevante Geschichtsdaten für Schulaufgaben und Tests in der Uhr abzulegen. All diesen Schülern, die an eine neue Ära des Spikens geglaubt haben, müssen wir leider eine traurige Mitteilung machen: die Uhr piept. Das bedeutet, beim Weiterschalten von einer Seite auf die andere ertönt ein hoher Piepston, ein verräterisches Signal für den Lehrer. Die Lösungsansätze der Gewinner zeichneten sich durch eine ausgefallene und dennoch praktikable Lösung der Ausnutzung des Speicherplatzes aus. Besonders hervorzuheben ist bei diesen Lösungen die Verwendung von sinnvollen Abkürzungsalgorithmen, die eine größtmögliche Datensammlung in der Uhr gewährleistet.

Die zehn Gewinner sind:

Rosemarie Muche, Berlin
Hans-Georg Troschke, Geilenkirchen
Hans Joachim Liesert, Münster
Axel König, Meinerzhagen
Dominique Gerhard, Luxembourg
Karl-Heinz Quader, Hamburg
Walter Geier, Wien
Oliver Mangold, Laichingen
Wolf-Dieter Wirz, Koblenz
Lutz Stohlmann, Braunschweig

Die Gewinner erhalten die Uhren mit Interface und Software im Wert von jeweils 450 Mark zugesandt.

Reparaturerfahrungen gefragt

Mit diesem Fragebogen schneiden wir ein Thema an, das sicherlich alle Computerbesitzer interessieren dürfte. Es kann nämlich jeden Augenblick passieren, daß der Computer in die ewigen Jagdgründe der Chips eingeht. Dann ist guter Rat und gute Reparatur oft teuer und zeitraubend. Wir möchten wissen, welche Erfahrungen Sie mit einem defekten C 64, VC 20 oder Floppy-Laufwerk gemacht haben. Helfen Sie mit, durch Ihre Angaben von der Reparatursituation in Deutschland eine Art Bestandsaufnahme zu erstellen. Sind die Mängel bekannt, so können von der 64'er Redaktion in Zusammenarbeit mit den betroffenen Firmen Schritte zu deren Verbesserung unternommen werden.

Jeder der mitmacht, die Reparaturlandschaft zu verbessern, hat die Chance einen Preis zu gewinnen. Der erste Preis besteht in einem Commodore 128 Personal Computer, die Preise 2 bis 11 sind professionelle Programme von Commodore.

Die Gewinner werden aus den Einsendungen ausgelost.

Der Rechtsweg ist ausgeschlossen.

Einsendeschluß ist der 15. August 1985.

(aa)

FRAGEBOGEN: Service- und Reparaturerfahrungen gefragt (Bitte ausfüllen und einsenden)



1. Welche Geräte besitzen Sie?

Geräte	Typ	seit (Monat/Jahr)
Computer		
Datasette		
Floppy-Laufwerk		
Drucker		
Monitor		
Joystick		

2. Wo haben Sie die Geräte gekauft?
- ☐ Commodore-Fachhändler
- ☐ Kaufhaus
- ☐ Versandhandel
- ☐ Andere Handelsform

Welche? ______________________

3. Wie beurteilen Sie die Beratung Ihres Händlers?
- ☐ gut
- ☐ ausreichend
- ☐ verbesserungswürdig

4. Wie könnte die Beratung verbessert werden?

____

____

____

____

____

____

____

5. Wieviel Stunden pro Woche benutzen Sie die Geräte?

__________ Stunden pro Woche

6. Anzahl der Reparaturen vor der Garantie und danach?

Geräte	Anzahl Störungen/Reparaturen vor Garantie	danach	Durchschnittl. Reparaturkosten
Computer			
Datasette			
Floppy-Laufwerk			
Drucker			
Monitor			
Joystick			

7. Welche Reparaturen traten auf?

Geräte	Schäden

8. Wie lange dauerten die Reparaturen?

Geräte	Dauer

9. Wie waren Sie mit dem Service zufrieden?
- ☐ gut
- ☐ ausreichend
- ☐ verbesserungswürdig

10. Was sollte verbessert werden?

____

____

____

____

____

____

____

____

11. Haben Sie Ihre Geräte schon selbst repariert?
- ☐ Ja

Welche Reparaturen? ______________________

____

____

____

____

- ☐ Nein

12. Eigene Einschätzung
- ☐ Anfänger
- ☐ Fortgeschrittener
- ☐ Profi

Absender:

Name: ______________________

Vorname: ______________________

Straße: ______________________

Ort: ______________________



Wir suchen die Anwendung des Monats

Anwendung des Monats, was ist das? Nun, Sie haben einen Commodore 64 oder einen VC 20 und versuchen diesen irgendwie sinnvoll einzusetzen. Unter einer sinnvollen Anwendung versteht die 64'er Redaktion alles, was beispielsweise Programme im häuslichen Bereich bewirken. Es kann sich dabei um die Berechnung der Benzinkosten für Ihren Wagen handeln, um ein eigenes Textverarbeitungsprogramm gehen, sich um die Verwaltung Ihrer Tiefkühltruhe drehen oder ein ausgeklügeltes Telefon- und Adreßregister sein.

Setzen Sie Ihren VC 20/C 64 mehr oder weniger beruflich ein? Auch, oder vor allem, das ist eine sinnvolle Anwendung. Sie führen die Lohn- und Gehaltsabrechnung, Ihre Lagerverwaltung, die Bestellungen auf einem Commodore-Heimcomputer durch? So spezielle Anwendungen wie die Berechnung der Statik von selbstgezimmerten Regalen, von Klimadiagrammen oder Vokabellernprogrammen für den Schulunterricht oder die Zinsberechnung bei Krediten sind ebenfalls Themen, die mehr als konkurrenzfähig sind.

Uns ist die Anwendung des Monats

500 Mark

wert.

Schreiben Sie uns, was Sie mit Ihrem Computer machen:

Redaktion 64'er, Aktion: Anwendung des Monats, Hans-Pinsel-Str. 2, 8013 Haar bei München.

Einmal im Monat gibt es die SUPERCHANCE

Diese nicht einmalige Gelegenheit sollten Sie nutzen. Wie? Schicken Sie uns Ihr bestes, selbst erstelltes Programm. Bei der Art des Programms sind wir nicht wählerisch.

Sie haben ein sehr gutes (Schieß-, Knobel-, Denk-, Action-, Abenteuer-)Spiel geschrieben: einschicken!

Sie verfügen über ein komfortables Disketten-Kopier-(Sortier) Programm mit einigen außergewöhnlichen Leistungsmerkmalen: einschicken!

Sie haben das Basic um einige sinnvolle Befehle erweitert: einschicken!

Sie arbeiten mit einem selbsterstellten Textverarbeitungsprogramm, einer eigenen Tabellenkalkulation, einem semiprofessionellen Datenverwaltungsprogramm: einschicken!

Sie zeichnen und konstruieren mit einem selbsterstellten Programm in hochauflösender Grafik: einschicken!

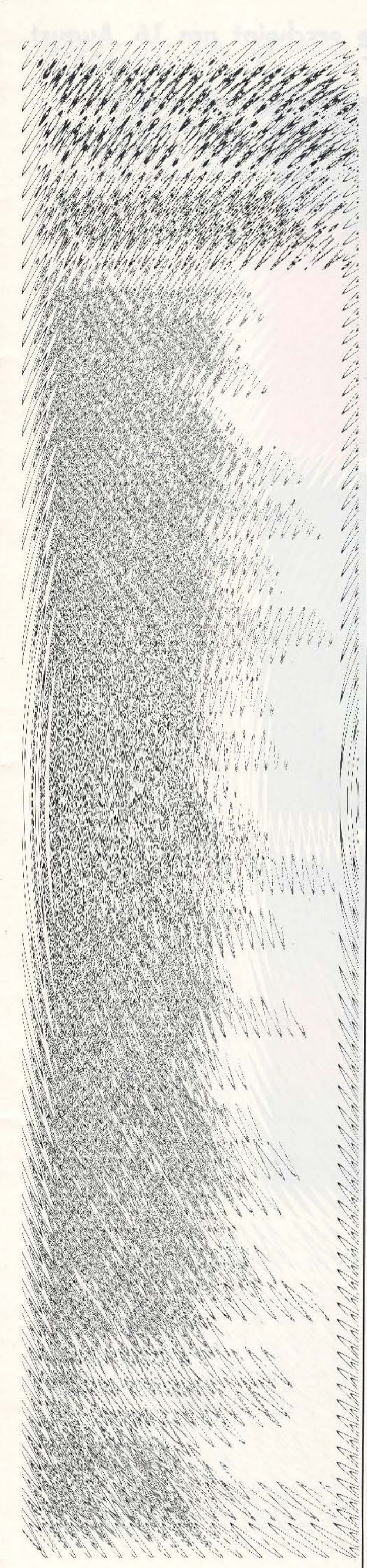
Wir freuen uns über jeden Beitrag und honorieren mit bis zu

2000 Mark

für das Listing des Monats

Aus den besten Listings, die veröffentlicht werden, sucht die 64'er-Redaktion einmal im Monat das »Listing des Monats« aus. Alle Listings, die im 64'er abgedruckt sind, werden mit 100 bis 300 Mark honoriert. Die genaue Vorgehensweise beim Einsenden von Listings ist in dem Beitrag »Wie schicke ich meine Programme ein?« in verschiedenen Ausgaben beschrieben.

Schicken Sie Ihr Listing an: Redaktion 64'er, Superchance: Listing des Monats, Hans-Pinsel-Str. 2, 8013 Haar bei München.



Inserentenverzeichnis

Impressum

Herausgeber: Carl-Franz von Quadt, Otmar Weber
Chefredakteur: Michael Scharfenberger (sc)
Redakteure: aa = Albert Absmeier, leitender Redakteur, ah = Achim Hübner, ev = Volker Everts, gk = Georg Klinge, hm = Harald Meyer, rg = Christian Rogge
Redaktionsassistenz: Gerda Vogl (202)
Fotografie: Janos Feitser/Jens Jancke, Titelfoto: Jens Jancke
Layout: Leo Eder (Ltg.), Dagmar Berninger, Willi Gründl
Auslandsrepräsentation:
Schweiz: Markt&Technik Vertriebs AG, Kollerstr. 3, CH-6300 Zug, Tel. 042-2231 55/56, Telex: 862329 mut ch
USA: M & T Publishing, 2464 Embarcadero Way, Palo Alto, CA 94303; Tel. (415) 424-0600; Telex 752351
Manuskripteinsendungen: Manuskripte und Programmlistings werden gerne von der Redaktion angenommen. Sie müssen frei sein von Rechten Dritter. Sollten sie auch an anderer Stelle zur Veröffentlichung oder gewerblichen Nutzung angeboten werden, so muß dies angegeben werden. Mit der Einsendung von Manuskripten und Listings gibt der Verfasser die Zustimmung zum Abdruck in von der Markt & Technik Verlags AG herausgegebenen Publikationen und zur Vervielfältigung der Programmlistings auf Datenträger. Honorare nach Vereinbarung. Für unverlangt eingesandte Manuskripte und Listings wird keine Haftung übernommen.
Herstellung: Klaus Buck (180)
Anzeigenverkaufsleitung: Ralph Peter Rauchfuss (126)
Anzeigenverkauf: Brigitta Fiebig (211)
Anzeigenverwaltung und Disposition: Michaela Hörl (171)
Anzeigenformate: ¹⁄₁-Seite ist 266 Millimeter hoch und 185 Millimeter breit (3 Spalten à 58 mm oder 4 Spalten à 43 Millimeter). Vollformat 297 x 210 Millimeter. Beilagen und Beihefter siehe Anzeigenpreisliste.
Anzeigenpreise: Es gilt die Anzeigenpreisliste Nr. 2 vom 1. Januar 1985.
Anzeigengrundpreise: ¹⁄₁ Seite sw: DM 8500,-. Farbzuschlag: erste und zweite Zusatzfarbe aus Europaskala je DM 1400,-. Vierfarbzuschlag DM 3800,-. Plazierung innerhalb der redaktionellen Beiträge: Mindestgröße ½-Seite
Anzeigen im Computer-Markt: Die ermäßigten Preise im Computer-Markt gelten nur innerhalb des geschlossenen Anzeigenteils, der ohne redaktionelle Beiträge ist. ¹⁄₁-Seite sw: DM 6400,-. Farbzuschlag: erste und zweite Zusatzfarbe aus Europaskala je DM 1000,-. Vierfarbzuschlag DM 3000,-. **Anzeigen in der Fundgrube: Private Kleinanzeigen** mit maximal 5 Zeilen Text DM 5,- je Anzeige.
Gewerbliche Kleinanzeigen: DM 11,- je Zeile Text.
Auf alle Anzeigenpreise wird die gesetzliche MwSt. jeweils zugerechnet.
Vertriebsleitung, Werbung: Hans Hörl (114)
Vertrieb Handelsauflage: Inland (Groß-, Einzel- und Bahnhofsbuchhandel) sowie Österreich und Schweiz: Pegasus Buch- und Zeitschriften-Vertriebsgesellschaft mbH, Hauptstätterstraße 96, 7000 Stuttgart 1, Telefon (0711) 6483-0
Erscheinungsweise: 64'er, Magazin für Computerfans, erscheint monatlich, Mitte des Vormonats.
Bezugsmöglichkeiten: Leser-Service: Telefon 089/4613-119. Bestellungen nimmt der Verlag oder jede Buchhandlung entgegen. Das Abonnement verlängert sich zu den dann jeweils gültigen Bedingungen um ein Jahr, wenn es nicht zwei Monate vor Ablauf schriftlich gekündigt wird.
Bezugspreise: Das Einzelheft kostet DM 6,50. Der Abonnementspreis beträgt im Inland DM 78,- pro Jahr für 12 Ausgaben. Darin enthalten sind die gesetzliche Mehrwertsteuer und die Zustellgebühren. Der Abonnementspreis erhöht sich um DM 18,- für die Zustellung im Ausland, für die Luftpostzustellung in Ländergruppe 1 (z.B. USA) um DM 38,-, in Ländergruppe 2 (z.B. Hongkong) um DM 58,-, in Ländergruppe 3 (z.B. Australien) um DM 68,-.
Druck: E. Schwend GmbH, Schmollerstr. 31, 7170 Schwäbisch Hall

Verantwortlich: Für redaktionellen Teil: Michael Scharfenberger.
Für Anzeigen: Brigitta Fiebig.
Redaktions-Direktor: Michael M. Pauly
Vorstand: Carl-Franz von Quadt, Otmar Weber
Anschrift für Verlag, Redaktion, Vertrieb, Anzeigenverwaltung und alle Verantwortlichen:
Markt&Technik Verlag Aktiengesellschaft, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon 089/4613-0, Telex 522052

Mitteilung gem. Bayerischem Pressegesetz: Aktionäre, die mehr als 25% des Kapitals halten: Otmar Weber, Ingenieur, München; Carl-Franz von Quadt, Betriebswirt, München. Aufsichtsrat: Dr. Robert Dissmann (Vorsitzender), Karl-Heinz Fanselow, Eduard Heilmayr.

Telefon-Durchwahl im Verlag:

Wählen Sie direkt: Per Durchwahl erreichen Sie alle Abteilungen direkt. Sie wählen 089-4613 und dann die Nummer, die in Klammern hinter dem jeweiligen Namen angegeben ist.

Mitglied der Informationsgemeinschaft zur Feststellung der Verbreitung von Werbeträgern e.V. (IVW), Bad Godesberg.



Super-Spiel zum Abtippen

Eine besondere Art des effektvollen Scrolling zeichnet dieses Spiel aus. Nur ein Grund, es zum Listing des Monats zu küren. Mit einer sehr ausführlichen Beschreibung zeigen wir Ihnen, wie Sie dieses Scrolling selbst programmieren und in eigene Programme einbauen können.

Spielen durch die Post

»Play by Mail«, das Spielen durch die Post, wird in Deutschland immer beliebter. Jetzt entwickelt sich aus dieser Form des Spielens eine neue Variante: »Play by Modem«. Was sich dahinter verbirgt und wie Sie sich daran beteiligen können, erfahren Sie in unserem Bericht.

C-Compiler und Super-Forth

Zwei brandneue Programmiersprachen haben wir für Sie getestet. Die Programmiersprache C ist ähnlich wie Pascal sehr stukturiert aufgebaut. Sie ermöglicht aber auch derart maschinennahe Programme, daß auf größeren Computern bereits ganze Betriebssysteme in C geschrieben sind.

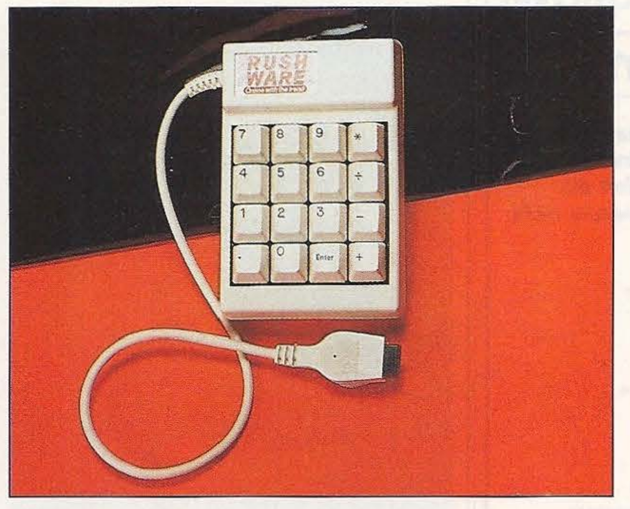
Super-Forth enthält Befehle zur Simulation künstlicher Intelligenz. Beispielsweise der nichtnumerischen Auflösung mathematischer Formeln.

Startext

Textverarbeitung ist mit einem Computer eine feine Sache. Doch wer ein gutes Programm haben will, muß schon etwas tiefer in die Tasche greifen. Leistung kostet seinen Preis. Daß es auch anders geht, beweist Startext. Für 64 Mark stellt Sybex ein Textprogramm der Spitzenklasse vor. Die Tester bekamen glänzende Augen. Ob es sich auch in der Praxis bewährt, erfahren Sie in der nächsten Ausgabe.

Außerdem...

— Neue Sonder-Poster zum Raustrennen
— Ein Wettbewerb zur Spielelandschaft mit tollen Preisen
— Das neue Copyrightgesetz und seine Auswirkungen
— und wieder viele Tips und Tricks für den C 64 und VC 20



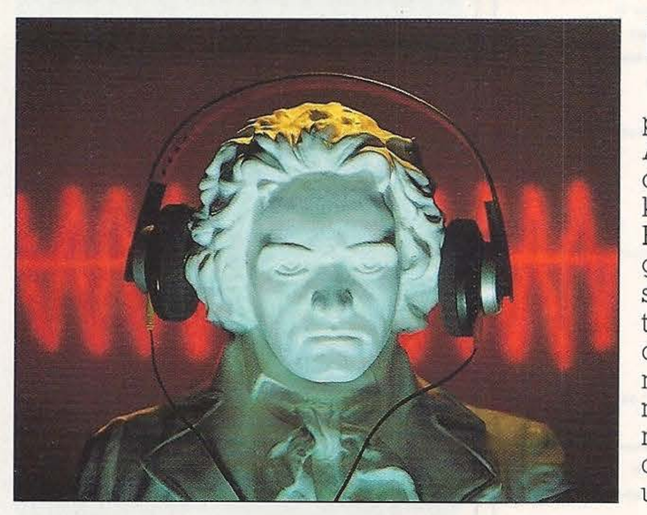
Alle Eingabegeräte für den Commodore 64

Wir stellen Ihnen die verschiedenen Eingabegeräte für den C 64 vor: Tastatur, Joystick, Maus & Co.

In einem ausführlichen Artikel sagen wir Ihnen wie sie funktionieren und wozu man sie am sinnvollsten einsetzt.

Basic-Erweiterungen für Grafik

Wollen Sie Grafiken in eigene Programme aufnehmen, kommt Ihnen der Befehlsvorrat des C 64 nicht gerade entgegen. Wir untersuchen Basic-Erweiterungen und nehmen speziell die Leistungen der Grafikbefehle unter die Lupe.



Der Musikus

Einer der Schwerpunkte der nächsten Ausgabe ist Musik mit dem C 64. Sie erhalten konkrete Antworten auf Fragen zur Musikprogrammierung, die Sie sicher schon immer interessiert haben. Nach dieser Ausgabe können Sie Ihre Programme mit den Klängen eines Streichorchesters oder einer Punkband unterlegen.

Sportkassenverwaltung

Immer wieder tauchen in Kassenberichten Fehler und Unstimmigkeiten auf, wenn sie handschriftlich erstellt werden. Zeitraubendes Neuschreiben und Ändern ist dann unumgänglich. Dieses Programm macht alles viel leichter. Fehler lassen sich schnell erkennen und beheben, zudem erleichtert die Gliederung der gedruckten Abrechnung die Jahresabrechnung erheblich. Die Anwendung des Monats für den C 64 und den VC 20.

Test: Drucker

Wir stellen Ihnen drei Drucker in ausführlichen Testberichten vor. Der eine ist der Panasonic KX-P1091, der ein sehr gutes Preis-/Leistungsverhältnis verspricht. Beim anderen handelt es sich um eine neue Version des Star SG-10C, der einen deutschen Zeichensatz bekam. Der dritte schließlich ist der Riteman C+ mit einem neuartigen Papiereinzugsystem, das den Platzbedarf des Riteman gegenüber anderen Druckern um einiges reduziert.



